NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ENCOUNTER DETECTION USING VISUAL ANALYTICS TO IMPROVE MARITIME DOMAIN AWARENESS

by

Michael J. Hanna

June 2015

Thesis Advisor:	David A. Garren
Thesis Co-Advisor:	James W. Scrofani
Second Reader:	Steven E. Pilnick

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2015	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
ENCOUNTER DETECTION USING VISUAL ANALYTICS TO IMPROVE MARITIME DOMAIN AWARENESS

5. FUNDING NUMBERS

6. AUTHOR(S) Michael J. Hanna

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

A visual analytics process to detect encounters between vessels from ship position data is developed in this thesis. An archive of historical position records is pre-processed and filtered to provide input for an encounter detection algorithm. The algorithm arranges the position records into a set of sorted lists (SSL) so that only a minimum number of records need to be compared. The algorithm performs a single sweep over the record set to arrange it into a SSL and simultaneously find the encounters. To avoid problems due to discrete sampling, an interpolation of the data is performed when the sampling is too sparse. To accommodate large data sets, a divide-and-conquer approach using a sliding spatial window is developed. In post-processing, the elementary encounters are grouped into composite encounters by collecting elementary encounters occurring between the same vessels. Additionally, the composite encounters are input into a visual analytics tool where each composite encounter is represented as a layer on a map. Patterns of life analysis and investigations of potential anomalous activity are performed by zooming in on encounter areas of interest. The development of a visual analytics process to identify vessels of interest is the significant result of this thesis.

14. SUBJECT TERMS maritime domain awareness, encounter detection, visual analytics, activity based intelligence

15. NUMBER OF PAGES 143

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ENCOUNTER DETECTION USING VISUAL ANALYTICS TO IMPROVE MARITIME DOMAIN AWARENESS

Michael J. Hanna
B. S., California Polytechnic University, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2015

Author: Michael J. Hanna

Approved by: Dr. David A. Garren
Thesis Advisor

Dr. James W. Scrofani
Thesis Co-Advisor

Dr. Steven E. Pilnick
Second Reader

Dr. R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A visual analytics process to detect encounters between vessels from ship position data is developed in this thesis. An archive of historical position records is pre-processed and filtered to provide input for an encounter detection algorithm. The algorithm arranges the position records into a set of sorted lists (SSL) so that only a minimum number of records need to be compared. The algorithm performs a single sweep over the record set to arrange it into a SSL and simultaneously find the encounters. To avoid problems due to discrete sampling, an interpolation of the data is performed when the sampling is too sparse. To accommodate large data sets, a divide-and-conquer approach using a sliding spatial window is developed. In post-processing, the elementary encounters are grouped into composite encounters by collecting elementary encounters occurring between the same vessels. Additionally, the composite encounters are input into a visual analytics tool where each composite encounter is represented as a layer on a map. Patterns of life analysis and investigations of potential anomalous activity are performed by zooming in on encounter areas of interest. The development of a visual analytics process to identify vessels of interest is the significant result of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ABI	activity-based intelligence
AIS	automated information system
COP	common operating picture
CSV	Comma -separated values
CWS	Coast Watch System
DHS	Department of Homeland Security
EDA	exploratory data analysis
GIS	geographic information science
GNSS	global navigation satellite system
GPS	global positioning system
HUMINT	human intelligence
IC	Intelligence Community
KD	knowledge discovery
MA	machine analytics
MASINT	measurement and signatures intelligence
MATLAB	Matrix Laboratory
MDA	maritime domain awareness
MMSI	Maritime Mobile Service Identity
MPO	Moving Point Object
NGA	National Geospatial-Intelligence Agency
NMEA	National Maritime Electronics Association
NRO	National Reconnaissance Organization
NSA	National Security Agency
NVAC	National Visualization and Analytics Center
OUSDI	Office of the Undersecretary of Defense for Intelligence
SIGINT	signals intelligence
SPIE	Society of Photo-Optical Instrumentation Engineers
SSL	set of sorted lists
TBA	tri-border area

USN	United States Navy
VA	visual analytics

EXECUTIVE SUMMARY

Activity-based intelligence (ABI) is a recent development in Intelligence Community (IC) tradecraft for dealing with the ever-increasing volumes of data collected. ABI represents a paradigm shift from targeted collection efforts to a focus on discovery of unknown activities [1]. The objective of ABI is to provide analysts with an understanding of patterns of behavior and to assist analysts with initiating new collection activities or refining existing collection efforts [2].

ABI tradecraft focuses on the activities and transactions associated with an entity, a population, or an area of interest [3]. Analysts use data and the related metadata to identify associations within the data. Patterns of life can be determined using large collections of association data and its metadata [4, 5].

Maritime domain awareness (MDA) is a United States Navy (USN) concept for fusing intelligence with situational awareness, which includes all vessels, cargo and people related to a sea, ocean or other navigable waterway [6]. MDA requires the ability to monitor activities and transactions of entities to identify trends and anomalies in behavior. The MDA concept addresses many potential threats to maritime security including nation-state threats, terrorist threats, criminal and piracy threats, and environmental and social threats.

Visual Analytics (VA) is a multi-disciplinary research field that focuses on analytical techniques, data transformations, visual representations and user interactions. VA research develops new and innovative ways to combine knowledge from data mining and information visualization with user interfaces to create processes that combine computer processing with human analysis [7].

In this thesis, ABI tradecraft is developed to improve MDA. The ABI tradecraft is developed by applying recent VA research in movement to the maritime domain. A significant result of this thesis research is a VA process to detect maritime encounters between vessels. An encounter detection algorithm is demonstrated using historical positions records from an archive of Automated Information System (AIS) data. To

accommodate large data sets, a divide and conquer approach using a sliding spatial window is developed. Post-processing is used to group elementary encounters into composite encounters. The composite encounters are viewed as layers on a map in a visual analytics tool. Patterns of life analysis and investigations of potential anomalous activity are performed by zooming in on encounter areas of interest.

The encounter detection algorithm contains three key elements. The first element is scalability of the algorithm for the number of vessels, position records, and the temporal and spatial resolution of the data. The second element is efficiency of the algorithm in building the data structure and detecting elementary encounters. The third element is interpolation of the data to create additional position records.

The ABI tradecraft developed in this thesis was applied to vessels operating in the tri-border area (TBA) between the Philippines, Malaysia and Indonesia. The TBA was chosen as a geographic area of interest based on a history of piracy in the region. The TBA constitutes a single geopolitical space whose security influences the entire Southeast Asia maritime domain.

An archive of AIS data was explored in this thesis in order to identify potential vessels of interest (VOIs) and to investigate possible patterns of life in the maritime domain. The historical position records were pre-processed using AisDecoder software. The encounter detection algorithm was implemented in MATLAB, and the composite encounters were investigated using the geographic visualization environment in the V-Analytics software.

LIST OF REFERENCES

[1] C. Johnston (2013). *Modernizing defense intelligence: object based production and activity based intelligence* [PowerPoint slides]. Retrieved from https://info.publicintelligence.net/DIA-ActivityBasedIntelligence.pdf," 2013.

[2] D. Meyerriecks (2012). *Empowering intelligence integration: the (future) role of ground* [PowerPoint slides]. Retrieved from http://gsaw.org/wp-content/uploads/2013/07/2012s08meyerriecks.pdf

[3] M. Phillips, (2012, Sept. 28). A brief overview of activity based intelligence and human domain analytics, Trajectory [Online]. http://trajectorymagazine.com/defense-intelligence/item/1369-human-domain-analytics.html

[4] K. L. Barber, "NSG expeditionary architecture: harnessing big data," *National Geospatial-Intelligence Agency Magazine: Pathfinder,* vol. 10, no. 5, pp. 8–10, Sept./Oct. 2012.

[5] T. D. Lash, "Uses of motion imagery in activity-based intelligence," *Proc. SPIE,* 2013, vol. 8740, 874005 (May 16, 2013).

[6] U.S. Department of Homeland Security (DHS), "National plan to achieve maritime domain awareness for the national strategy for maritime security," DHS, Washington, DC, Oct. 2005.

[7] J. J. Thomas and K. A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* Los Alamitos, CA: IEEE, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The Office of the Undersecretary of Defense for Intelligence (OUSDI) defined Activity-based intelligence (ABI) as "a discipline of intelligence where the analysis and subsequent collection is focused on activity and transactions associated with an entity, population, or area of interest" [1, 2]. ABI is a recent development in tradecraft with a focus on discovery and association of unknown activities [3]. ABI provides understanding for patterns of behavior that enable analysts to initiate new collection activities or refine existing efforts [4].

Analysts develop ABI by identifying associations within the data. Analysts use activity data and the related metadata to identify entities of interest and develop patterns of life [1, 2]. ABI consumes volumes of data in order to characterize and relate activities and transactions [5].

The tradecraft of ABI is applied in this thesis to the United States Navy (USN) concept of maritime domain awareness (MDA). MDA is "the effective understanding of anything associated with the global maritime domain that could impact the security, safety, economy, or environment of the United States" [6]. The maritime domain is defined as "all areas and things of, on, under, relating to, adjacent to, or bordering on a sea, ocean, or other navigable waterway, including all maritime-related activities, infrastructure, people, cargo, and vessels and other conveyances" [6].

The USN MDA concept relies upon the fusion of intelligence with situational awareness. The observable information for vessels, people, facilities, cargo, infrastructure, sea lanes, threats, friendly forces and weather are the inputs to a Collect, Fuse, Analyze and Disseminate data flow process shown in Figure 1 [7].

The research for this thesis utilizes a historical archive of position records derived from the Automated Information System (AIS) and the Global Positioning System (GPS) [8]. The position records used in this research were collected by two commercial satellite corporations that archive and sell AIS data [9]. ORBCOMM and exactEarth corporations

collect AIS position reports by employing low Earth orbit satellites to collect reports globally.

Since there is very limited information available to the public about ABI, recent research in visual analytics (VA) is incorporated to develop ABI tradecraft in support of the USN MDA concept. VA is defined by the National Visualization and Analytics Center (NVAC) as "the science of analytical reasoning facilitated by interactive visual interfaces" [10].

A visual analytics process in support of the USN MDA concept is developed by applying recent VA research in movement and the maritime domain. The VA process detects elementary encounters between maritime vessels using an archive of AIS data. Post-processing is used to group elementary encounters into composite encounters. Composite encounters are investigated using a visual geographic environment.

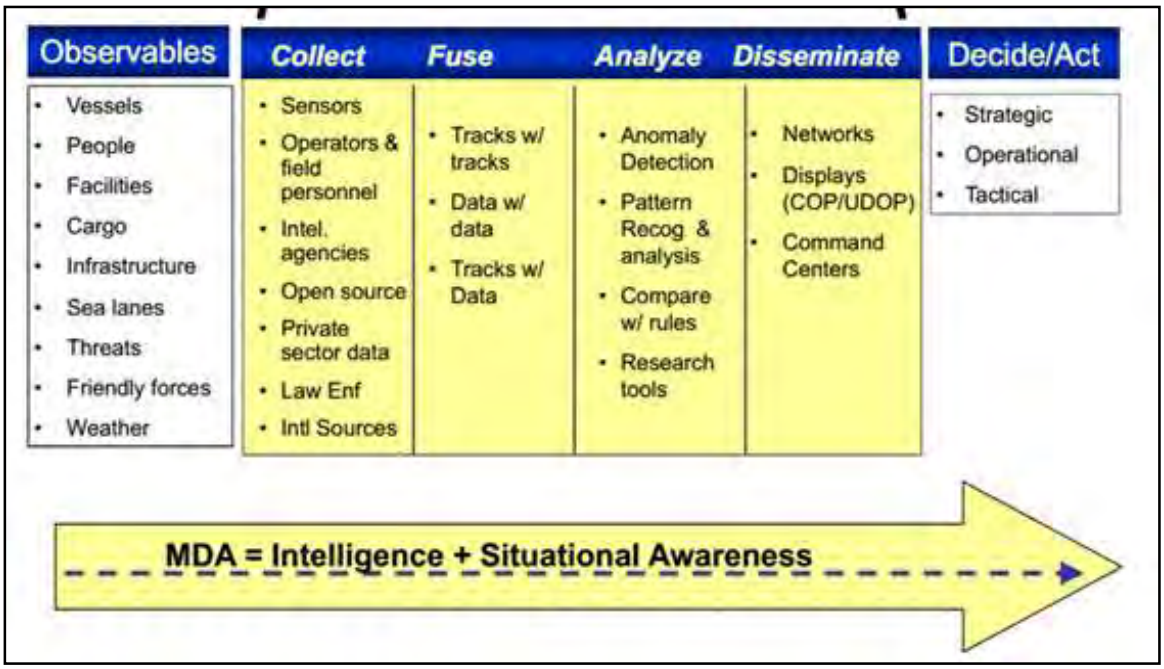

Figure 1. Maritime domain awareness data flow, from [7].

A. THESIS OBJECTIVE

In this thesis, ABI tradecraft is developed in support of MDA. The objective of this thesis is to develop a VA process to support the USN MDA concept. A recently published encounter detection algorithm from [11] is incorporated into a VA process.

The VA process consists of the following activities. Preprocessing of position records is performed using AisDecoder software. The encounter detection algorithm is executed using MATLAB. Post-processing in MATLAB is used to group elementary encounters to into composite encounters. Encounters are investigated using the geographic visualization environment in V-Analytics.

B. RELATED WORK

The related work for this thesis is previous research in MDA. Recent work includes previous thesis research performed at the Naval Postgraduate School (NPS). Analytical techniques and visual representations of vessel interactions were investigated by Tester in [8]. Tester developed a spatiotemporal algorithm using a *k*-means proximity filter to create a kinematic clustering of vessels based on similarity in course, speed, and distance. Techniques for anomaly detection in MDA were the focus of McAbee in [9]. McAbee developed a Hough transformation to generate atlases to represent normal vessel traffic patterns.

The application of VA principles to the maritime anomaly detection problem was the focus of Riveiro and a research team [12] at Orebro University in Sweden. Field research at three different maritime control centers was performed in order to understand the human factors requirements for a maritime anomaly detection system. A prototype system known as VISAD was developed that automatically provides an operator with visual models for normal vessel behavior.

Multi-sensor data fusion work in MDA includes the development of sensor data fusion architecture as described in [13]. An object-oriented sensor data fusion architecture that combined tracking data from video, radar and AIS sources was developed to create visualizations for situational awareness.

Maritime vessel traffic was investigated using density maps in [14]. Density maps were applied using a time-varying kernel to visualize vessel traffic patterns based on time of day and also to detect anomalous vessels operating outside of normal traffic patterns.

Tominski et al. proposed a novel visualization technique for viewing spatio-temporal data in [15]. The principle involved the use of stacked trajectory bands to provide a temporal representation of attribute values on a map and the introduction of a time lens to view the detailed temporal data. A maritime example included a comparison in tortuosity, the rate of change of headings, between incoming and outgoing vessels near ports.

Researchers in Canada performed a comprehensive review of VA literature and tools to address the need for a Recognized Maritime Picture (RMP) [16]. In addition to an annotated bibliography of 146 references, the review also provides a profile of the VA research community and supporting organizations.

The Embedded Systems Institute conducted project, known as POSEIDON, on maritime domain awareness from June 2007 to May 2012 as described in [17]. The POSEIDON project developed an architectural framework and prototype systems of systems (SoS) to improve maritime domain awareness. Additionally, the POSEIDON project developed methods for anomaly detection by mining vessel trajectory data.

A literature review of foundational technologies for ABI was reported in [18]. This literature review provides a survey of algorithmic-level literature that has been applied to ABI analysis.

C. THESIS OUTLINE

Background information on ABI, VA, information visualization and data mining and algorithms are included in Chapter II. The encounter detection algorithm and its implementation in MATLAB are developed in Chapter III. The VA process and the investigation of an archive of AIS data are described in Chapter IV. Considerations for future research and a summary of the research performed for this thesis are included in Chapter V. The MATLAB code developed for this thesis research is included in the Appendix.

II. BACKGROUND

Research for this thesis applies the concepts of ABI and VA to the USN MDA concept. The principles of ABI that were introduced in the previous chapter are discussed in the first section of this chapter. The history and concepts of VA that were also introduced in the previous chapter are discussed in the second section. An introduction to information visualization is provided in the third section, and an introduction to data mining, machine learning and algorithms is provided in the fourth and final section of this chapter.

A. ACTIVITY-BASED INTELLIGENCE

The Office of the Undersecretary of Defense Intelligence (OUSDI) defined the tenets of ABI in 2010 with the release of the first two strategic guidance papers [19]. The papers introduced ABI as a new discipline of intelligence that focuses analysis and subsequent collection on the activity and transactions associated with an entity, a population, or an area of interest.

With ABI, intelligence collection has shifted its focus from specific military targets to the actions and movements of individuals [19]. ABI has generated interest throughout the Intelligence Community (IC) and is a "natural evolution" that has taken place since the Cold War, according to Robert Zitz, a former senior executive with the National Geospatial-Intelligence Agency (NGA), National Security Agency (NSA) and the National Reconnaissance Organization (NRO) [20].

> On May 1, 2013, during the Society of Photo-Optical Instrumentation Engineers (SPIE) symposium of Defense, Security and Sensing, Director Letitia Long, said the NGA is using ABI to "identify patterns, trends, networks and relationships hidden within large data collections from multiple sources: full-motion video, multispectral imagery, infrared, radar, foundation data, as well as Signals Intelligence (SIGINT), Human Intelligence (HUMINT) and Measurement and Signatures Intelligence (MASINT) information." [21]

There are similarities between ABI principles and the USN MDA concept. MDA analysts and maritime operation centers (MOC) persistently monitor, access, and

maintain information on vessels, cargo, crew and passengers [8]. The MOC collects, fuses, analyzes and disseminates intelligence information; however, the ABI concept is larger than the USN MDA concept.

The principles of ABI can be summarized by the following five elements:

1. Collect, characterize and locate activities and transactions.
2. Identify and locate actors and entities conducting the activities and transactions.
3. Identify and locate networks of actors.
4. Understand the relationships between networks.
5. Develop patterns of life. [19]

ABI tradecraft brings the focus on entities and their activities and transactions to the forefront of analytical processes [5]. As explained by Phillips [19]: "The intention of ABI is to develop patterns of life, to determine which activities and transactions are abnormal, and to seek to understand those patterns to develop courses of action. ABI is focused on understanding relationships between various entities and their activities and transactions."

ABI is focused on resolving entities based on analysis of activities and transactions [5]. Activities and transactions for entities are characterized by analyzing large collections of data. An activity is an expression of change such as movement. A transaction is the exchange of something.

In this section, an overview of ABI tradecraft was provided. For a more detailed study of ABI tradecraft refer to [18]. In the next section, a brief history and the principles of VA are discussed.

B. VISUAL ANALYTICS

Before introducing the research agenda for VA, a brief overview of intelligence collection is provided. The changing requirements for intelligence collection are at the forefront of the VA research agenda and also led to the establishment of the ABI tradecraft discussed in the previous section.

1. Intelligence Collection

The results of a study published in 2004 [22] addressed two related problems in intelligence collection:

1. The inability to identify and retain all potentially useful data that is collected due to a lack of understanding of future utility and the need to reduce the overall massive volume of data.
2. The inability to find relevant data when viewed in a future context that was not understood when initially collected.

The study incorporated two techniques from commercial fraud detection to improve the detection of rare events and to create strong supporting evidence from the aggregation of otherwise weak data. The first technique was the use of historical data to investigate multiple hypotheses. The second technique was using results to refine existing collections or to initiate new collections.

The study also characterized the information processing data flow from the perspective of the analyst. The traditional data processing flow for intelligence collection is a "detect and track" scheme for the following process:

1. Filter all information being gathered, and save only the information that is needed for the immediate problem.
2. Detect individual facts in the data (i.e., people, places, things and simple relationships between them).
3. Process the results with queries built to uncover "gold nuggets" of intelligence information.
4. Track and report changes in the status of the information. [22]

An approach that inverts the traditional data flow is concept-based information processing [22]. Concept-based information processing uses a "track before detect" scheme with the following process:

1. Catalog and save all information being gathered based upon individual facts in the data.
2. Populate many parallel hypotheses with the facts.
3. Detect when hypotheses collect enough supporting facts to cross a threshold and report the fact.

4. Continue to aggregate evidence along the hypotheses, and report any changes.

5. Adapt and refine initial concepts. [22]

2. Research Agenda

New approaches to information processing were also a focus of the U.S. Department of Homeland Security (DHS). In 2004, the U.S. DHS chartered the National Visualization and Analytics Center (NVAC) to define a long-term research agenda for VA [10]. The research agenda for VA was developed under the leadership of the Pacific Northwest National Laboratory (PNNL).

VA is "the science of analytical reasoning facilitated by interactive visual interfaces" [10]. Analysts use VA tools and techniques to gain insight from massive data sets. VA tools and techniques help detect the expected and discover the unexpected.

VA is a multi-disciplinary research field with the following focus areas:

1. Analytical reasoning techniques to directly support decision making.

2. Visual representations and interaction techniques to explore large amounts of information at once.

3. Data representations and transformations that support visualization and analysis.

4. Techniques to support production, presentation and dissemination of results. [10]

VA aims to create software systems to support the analytical reasoning process [10]. The analytical process consists of collecting and organizing information towards judgment about a question. Throughout the process, the analyst identifies or creates pieces of information that contribute to reaching a defensible judgment. The pieces of information are referred to as reasoning artifacts. A summary of common reasoning artifacts is shown in Table 1.

VA research is focused on improving analytic discourse. Analytics discourse is "the technology-mediated dialogue between an analyst and his or her information to produce a judgment about an issue [10]."

The analyst's information consists of the following:

1. Understanding of the question being answered.
2. Information gathered which may or may not be relevant.
3. Analyst's knowledge including assumptions, hypotheses and arguments. [10]

Analytic discourse is the process to assemble evidence and assumptions on the path to articulate a judgment [10]. VA research seeks to leverage both the strength of the computer processing and the human analyst to improve analytic discourse. VA tools support analytic discourse for data retrieval, navigation and discovery.

3. Related Research Efforts

Prior to 2005 and the research agenda for VA, the concepts for VA were seen in research efforts in information visualization, geographic information science (GIS), geovisualization and data mining [23]. GIS is research in graphical representations of geographic information, while research in geovisualization aims to speed up the graphical display of geographic information using novel maps.

Previous research in GIS and geovisualization provides a foundation for current research in VA. Current research in VA is focused on the following principles:

1. Emphasis on data analysis, problem solving, and/or decision making.
2. Leveraging computational processing by applying automated techniques for data processing and knowledge discovery.
3. Active involvement of a human in the analytical process through interactive visual interfaces. [23]

A recent focus of the VA research community is VA methods for spatial-temporal information. A research agenda to address the challenges in working with space and time was published in 2009 [23]. One of the challenges for working with space and time is that dependencies in observations prohibit the use of standard statistical analysis, which assumes independence among observations. Spatial and temporal dependence, however, provide opportunities for data processing and analysis. Spatial and temporal dependence allow for the following data operations:

Table 1. Common reasoning artifacts, from [10].

Elemental artifacts: artifacts derived from isolated pieces of information	
Source Intelligence	An individual piece of intelligence (e.g., a document, photograph, signal, sensor reading) that has come to the analyst's attention through a collection or retrieval activity.
Relevant Information	Source intelligence that is believed to be relevant to the issue and usable for constructing arguments and judgments.
Assumption	An asserted fact, and its basis, that will be used for reasoning. Assumptions must be managed separately from evidence, as sound practice demands their critical inspection. An assumption may come from the analyst's prior knowledge, an earlier conclusion or product of an analysis, or a key, presently unknowable presumed fact that allows judgment to progress despite a gap in knowledge.
Evidence	The information or assumption takes on argument value when the analyst assesses its quality, accuracy, strength, certainty, and utility against higher-level knowledge artifacts such as hypotheses and scenarios. Assessing the utility can be as simple as judging if the evidence is consistent or inconsistent with a hypothesis or scenario or if the evidence argues for or against an inference.
Pattern artifacts: artifacts derived from collections of information	
Patterns and Structure	Relationships among many pieces of data to form evidence. Analysts often create tables, charts, and networks of data to detect and extract pattern or structure.
Temporal and Spatial Patterns	Temporal relationships and spatial patterns that may be revealed through timelines and maps. Changes in pattern, surprising events, coincidences, and anomalous timing may all lead to evidence recognition. The simple act of placing information on a timeline or a map can generate clarity and profound insight.
Higher-order knowledge constructs	
Arguments	Logical inferences linking evidence and other reasoning artifacts into defensible judgments of greater knowledge value. Extensive formal systems, such as predicate calculus, give a solid inferential basis.
Causality	Specialized inference about time, argument, and evidence that makes the argument that an event or action caused a second event or action. Causality is often critical to assessments. It is also a source of many biases and errors, and demands careful review.
Models of Estimation	A means of encoding a complex problem by understanding logic and applying it to evidence, resulting in a higher-level judgment that estimates the significance of available evidence to the issue at hand. Some important classes of models are utility models (which estimate the value of a potential action to an actor using multiple weighted criteria), indicator models (used to estimate if outcomes of interest may be in the process of development), behavioral models (of individual and group dynamics), economic models, and physical models. Specialized analytic activity may involve research using models, simulation, and gaming. A repertoire of basic problem modeling and structuring techniques is invaluable to the analyst.
Complex reasoning constructs	
Hypothesis	A conjectured explanation, assessment, or forecast that should be supported by the evidence.
Scenarios or Scenario Fragments	Sequences of information with "story" value in explaining or defending part of a judgment chain. For example, a threat scenario might address a target, method, actor, motive, means, and opportunity.

1. Interpolation and extrapolation to fill gaps in incomplete data.
2. Integration of information from multiple sources using references to common spatial-temporal information. [23]

VA research assumes that interactive visual representations of information can improve natural human capabilities for detecting patterns, making connections and creating inferences from data [23]; however, to validate the contributions of VA research to information processing, it is necessary to identify a way to measure the effectiveness of different approaches. An example effectiveness assessment involved presenting a user the same task but displaying the spatio-temporal information in three different ways.

Three commonly used but computationally different ways of displaying spatio-temporal information are:

1. A static small multiple map display.
2. A non-interactive animation.
3. An interactive animation with varying animation speeds. [23]

A study of users showed that users tested on a static small multiple map display seemed more focused on "states" and "spatial patterns" instead of events and time [23], while users with the animation displays focused on change and events instead of spatial patterns. Another finding from the user study was that novel interface tools are not necessarily used by the user to complete the task even though it would aid in the task.

Another challenge in working with spatio-temporal information is the processing and display of large data sets. A direct approach is to provide a visual depiction of each record in a data set within an interactive environment [23]. An alternative approach is to apply data aggregation and summarization methods to the data set prior to visualization. Another alternative is to apply more sophisticated computational methods to automatically extract specific patterns prior to visualization.

Despite the progress that has been made in VA research spatio-temporal data presents the following new challenges:

1. Large data sets.
2. Dynamic data in real-time.

3. Combining data of diverse types. [23]

4. Machine Analytics

VA research can also be identified as a category within machine analytics (MA). The analysis tasks of data mining and data fusion are supported by MA [24]. MA contains the fields of descriptive, predictive and prescriptive analytics. Descriptive analytics use historical and current data to analyze a situation. Predictive analytics infer future trends, behavior and events to support decision making. Prescriptive analytics determine courses of action using historical, current and projected information for a prescribed set of objectives requirements or constraints.

MA also contains, in addition to VA, the fields of scientific and information visualization [24]. Scientific visualization applies to data with natural geometric structures, while information visualization is for visualizing abstract data structures. Within the categories of MA, the research in VA is unique in its emphasis of user interaction for the collection, exploitation and dissemination of data.

C. INFORMATION VISUALIZATION

Information visualization is the application of visualization techniques to data. Visualization is "the communication of information using graphical representations" [25]. Visualization provides the ability to comprehend huge amounts of data [26]. Visualization provides both qualitative and quantitative visual representations of information [25]. Visualizations allow the previously hidden emergent properties of a data set to be perceived [26].

The visualization process consists of several stages. The first stage is the collection and storage of data [26]. Visualization can incorporate data from a wide variety of sources and may be simple or complex in its construction [25]. Visualization can be used for exploration of data, to confirm a hypothesis or to present results. The next stage is a pre-processing stage to transform the data so it can be easily manipulated [26]. The pre-processing stage can also include data reduction in order to focus on only certain aspects of the data. The third and final stage in the visualization process consists of mapping the pre-processed data to a visual representation. The visual representation can

also include a user interface to allow for the transformation of the mapping or to alter the view of the data for the user.

Visualization is often used as part of a larger process such as exploratory data analysis (EDA), knowledge discovery (KD) or visual analytics (VA) [25]. Visualization and analysis are both used to build models that represent or approximate the data set. Visualization is used to discover new knowledge and identify patterns, anomalies and trends.

Before introducing data mining in the next section, a brief overview of the KD process is presented here. The first step in the KD process is data integration, warehousing and selection [25]. This step involves identifying data sets for analysis and performing any filtering, sampling and aggregation prior to data mining. The next step is data mining where an algorithm analyzes the data set in order to produce a model. After data mining, the results are presented to the user using visualization.

D. DATA MINING, MACHINE LEARNING AND ALGORITHMS

Data mining is the process to create knowledge from a large collection of data [27]. Descriptive data mining characterizes properties of data, while prescriptive data mining performs induction to make predictions. There are five distinct functions that data mining performs:

1. Characterization and discrimination of data.
2. Frequent pattern mining for associations and correlations.
3. Classification and regression analysis of data.
4. Clustering analysis of data.
5. Anomaly detection in data. [27]

Data mining is most closely associated with machine learning. Machine learning investigates how computer performance can be improved based on large data sets [27]. Machine learning research focuses on the automatic recognition of patterns and intelligent decision making by computers. There are four types of machine learning:

1. Supervised learning from labeled data; same as classification.

2. Unsupervised learning for unlabeled data; same as clustering.
3. Semi-supervised learning use both labeled and unlabeled data.
4. Active learning involves the user in the learning process. [27]

Integral to any computational problem solving process is the use of computer algorithms [28]. A computer algorithm is "a set of steps to accomplish a task that is described precisely enough that a computer can run it." There are two key principles to using computer algorithms to solve computational problems:

1. Algorithm always produces a correct solution to the problem.
2. Algorithm uses computational resources efficiently. [28]

The primary resource that determines efficiency in algorithms is time. There are five factors that affect the amount of time an algorithm requires to complete:

1. Speed of the computer used for processing data.
2. Programming language in which the algorithm is implemented.
3. Compiler or interpreter that translates the code.
4. Skill of the programmer who writes the program.
5. Other activity taking place on the computer at the same time. [28]

These factors are for the case of an algorithm that runs entirely in computer memory on a single computer [28].

III. ENCOUNTER DETECTION

Encounter detection and the visual analytics process to create visualizations are the focus of the research performed for this thesis. Within this chapter, the encounter pattern is introduced along with an overview of movement analysis. A conceptual model for movement is presented along with an introduction to encounter detection. Following an introduction to the visualization of movement, we present the encounter detection algorithm and visual analytics process. A discussion of the spatial window approach to encounter detection is provided in the final section as a conclusion to this chapter.

A. ENCOUNTER PATTERN

The encounter pattern is an example of a motion pattern for a group of Moving Point Objects (MPOs) [29]. MPOs are used to represent a wide range of diverse phenomena including animals in habitat and migration studies, vessels in a maritime environment and agents simulating people for modeling crowd behavior. The motion of MPOs can be represented by a series of observations consisting of a triple of identification, location and time.

An encounter is defined by Laube [29] as an extrapolated meeting between two or more vessels within some range R. The encounter consists of a set of m MPOs at interval i with vectors intersecting within a range R of radius r. An approach proposed by Laube to detect encounters was to solve a geometric problem using a simple algorithm that can be solved in $O(n^4)$ time. The notation $O(n^4)$ time describes the asymptotic growth in run time due to growth in the number of inputs being processed is given by the number of input values n raised to the fourth power

An encounter "refers to objects being close to one another in space and time" [30]. Further classification of encounters can be accomplished using additional attributes of MPOs such as direction and speed. The classification of encounters is domain dependent, and encounter detection in different domains requires different spatial and temporal windows.

In this thesis, encounter patterns found in historical AIS data are investigated to develop patterns of life for the maritime domain. Developing patterns of life is a key principle in the development of ABI tradecraft. By using historical AIS data, we followed a "track before detect" approach using the encounter pattern as a hypothesis to be investigated. The process of using a visual and interactive environment for exploring encounters follows an approach similar to previous VA research in the maritime domain.

B. MOVEMENT ANALYSIS

MPOs are entities where the position changes over time as reported by Dodge in [31]. MPOs are considered to be moving points with paths through space and time that can be visualized and analyzed.

The encounter pattern is just one of many patterns classified by Dodge in [31]. Dodge classifies the encounter pattern as a spatio-temporal pattern within a parent group of compound patterns. Compound patterns and primitive patterns are the groups within a category referred to as generic patterns. Dodge's classification approach divides all patterns between a category of generic patterns and a category of behavioral patterns. Dodge's proposed classification of movement patterns is shown in Figure 2.

Movement can also be decomposed into a series of elements referred to as movement parameters [31]. The three major groups of movement parameters are shown in Table 2. The historical position records used in this research contain both the spatial primitive of position (x,y) and the temporal primitive of instance (t). The archive of historical position records contains the position of each MPO for a series of timestamps. The timestamps are recorded by the satellite's GPS as they collect the AIS messages received.

In the context of movement analysis an encounter refers to moving to and meeting at the same location [31]. The encounter pattern is a specific instance of a convergence pattern. A convergence pattern may differ from an encounter pattern because not all MPOs need to arrive at the same time. A visual representation of the encounter pattern used in movement analysis is shown in Figure 3.

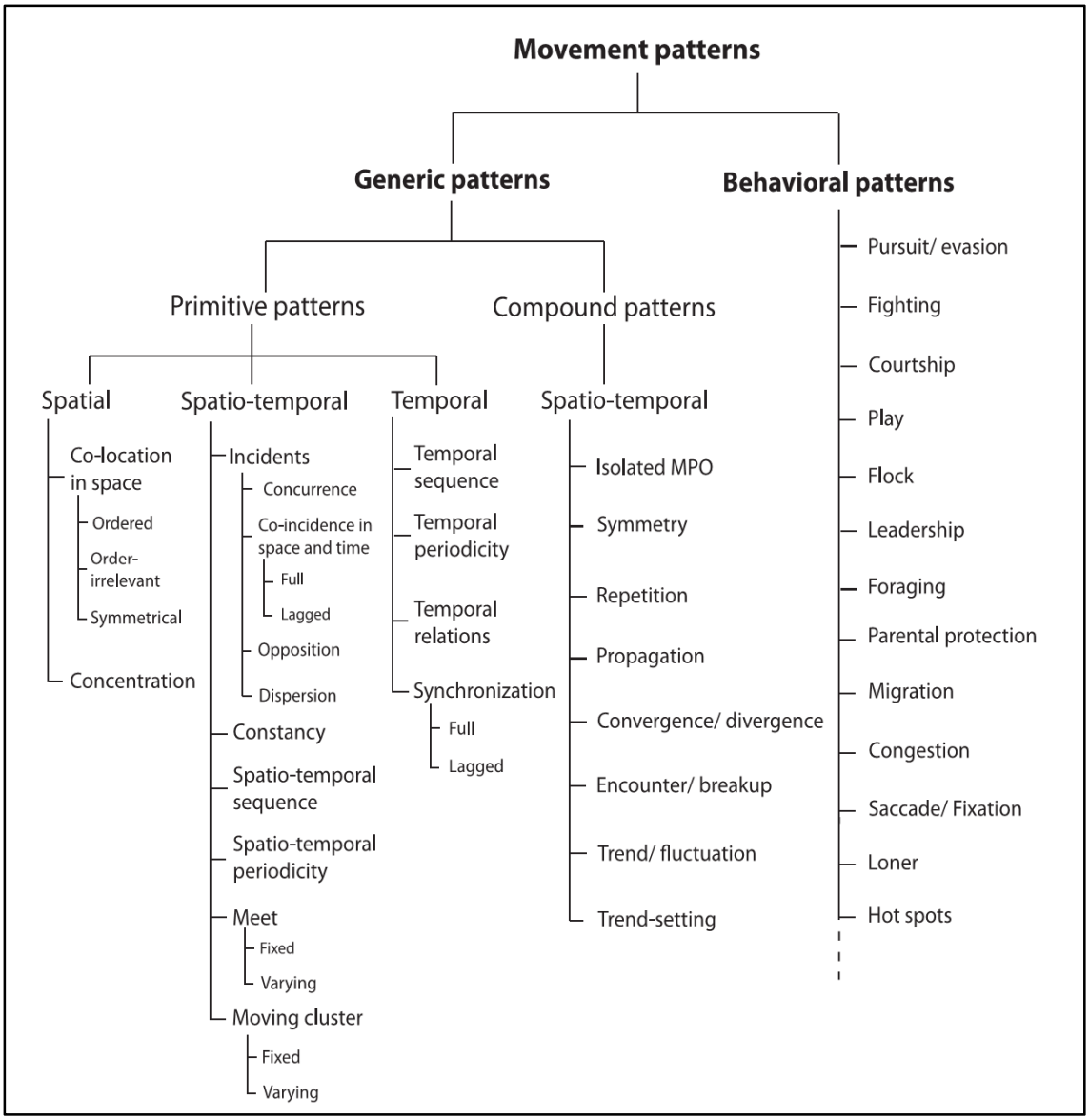

Figure 2. Classification of movement patterns, from [31].

Table 2. Movement parameters, from [31].

Parameters/Dimension	*Primitive*	*Primary derivatives*	*Secondary derivatives*
Spatial	Position *(x,y)*	Distance *f(posn)* Direction *f(posn)* Spatial extent *f(posn)*	Spatial distribution *f(distance)* Change of direction *f(direction)* Sinuosity *f(distance)*
Temporal	Instance *(t)* Interval *(t)*	Duration *f(t)* Travel time *f(t)*	Temporal distribution Change of duration *f(duration)*
Spatio-temporal *(x, y,t)*	—	Speed *f(x,y,t)* Velocity *f(x,y,t)*	Acceleration *f(speed)* Approaching rate

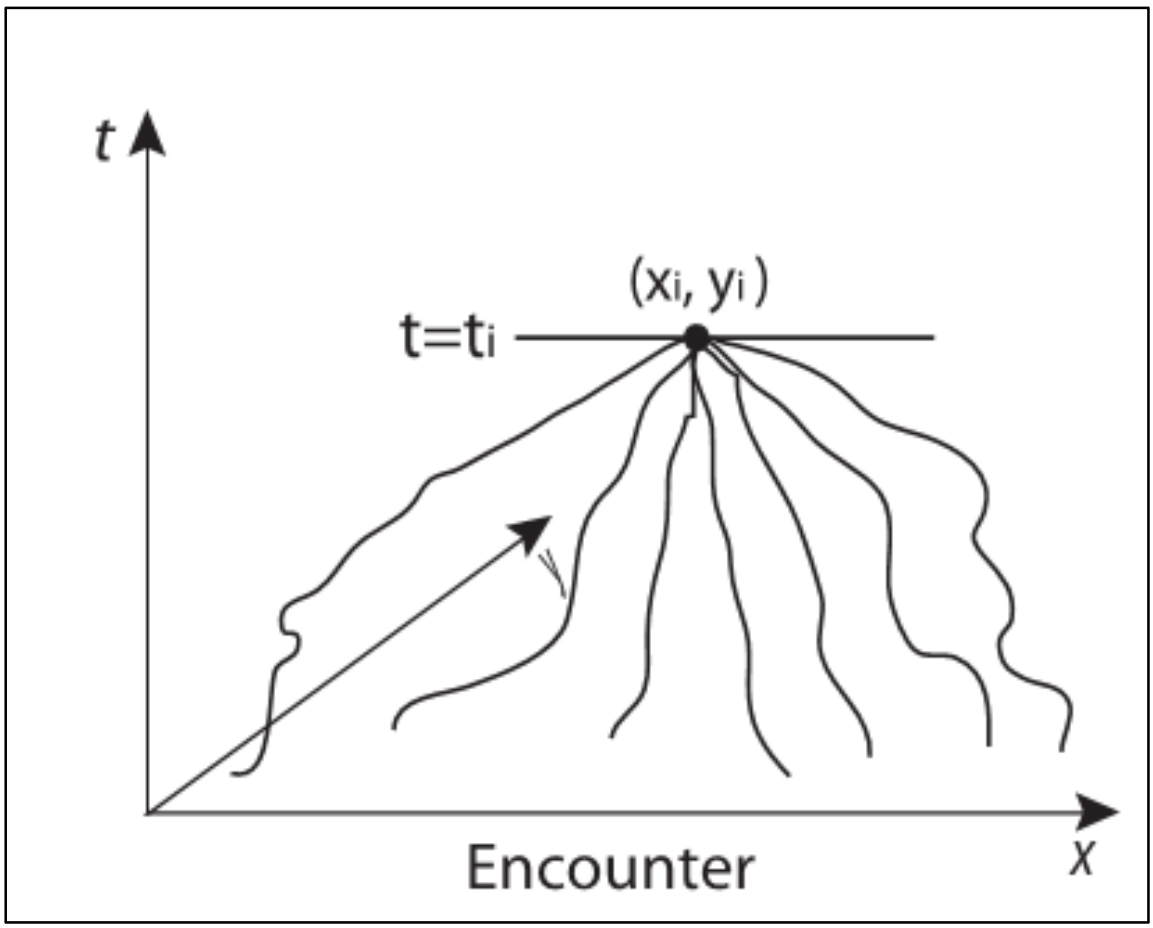

Figure 3. Encounter pattern, from [31].

C. CONCEPTUAL MOVEMENT MODEL

Behaviors in movement can only be understood by considering relations occurring between MPOs and the environment [32]. The environment also referred to as the spatio-temporal context includes the following:

1. Complex and heterogeneous physical space.
2. Complex and heterogeneous physical time.
3. Static and dynamic objects existing in space.
4. Events occurring over time. [32]

Visual representations of movement on a map allow a human analyst to see the relationship between movement and the spatial context [32]; however, maps do not

present temporal information unless the map is animated. The spatial track of an MPO can be represented by a trajectory [33]. Trajectories include all the positions occupied by the MPO between the start and end of movement.

Two visual representations of a trajectory are shown in Figure 4. On the left is an image showing the direction of trajectories using directional arrows. On the right is an image that additionally shows speed which is indicated by the width of the lines.

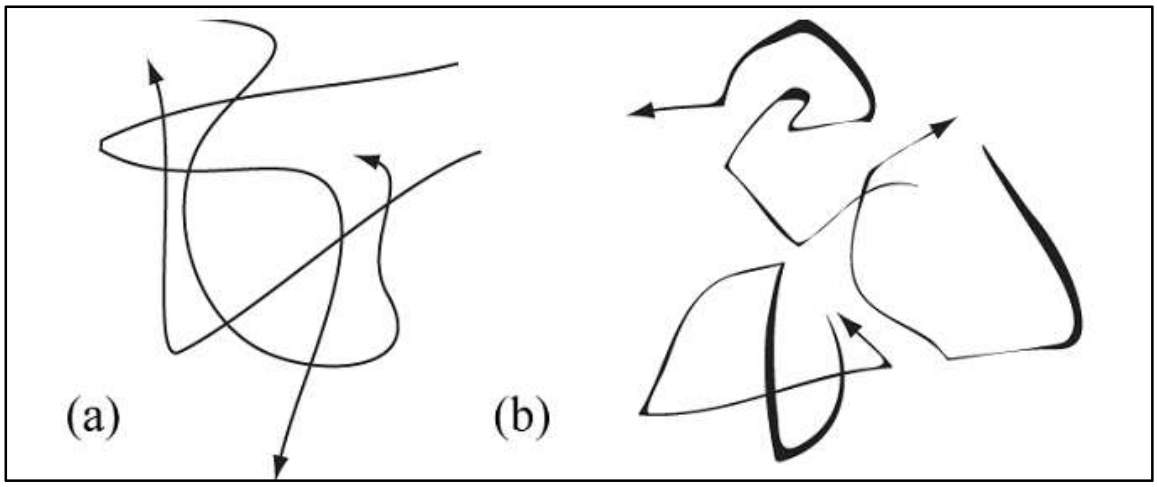

Figure 4. Visual representations of trajectories, from [33].

In defining a conceptual model there are three fundamental sets that are pertinent to movement: space *S* (set of locations), time *T* (set of instants), and objects *O* [32]. A classification of objects according to their spatial and temporal properties is shown in Table 3.

An encounter is a spatial event between two movers that have certain positions in space and in time. The encounter detection process identifies the encounter pattern when it occurs. Data mining is used to extract the encounter pattern from the large collection of data in the historical AIS archive. Trajectories are created for movers for the times when the encounter pattern is detected. In the case where more than two movers are involved in an encounter, additional trajectory pairs are defined until all have been accounted for.

Table 3. Classification of objects from, [32].

Concept	Superior concepts	Properties	Examples
Spatial object	Object	Has a certain position in space (a location or set of locations, not necessarily continuous)	Building, village, rainfall, deer, lynx, a deer at a river, a lynx chasing a deer
Event (temporal object)	Object	Appears and/or disappears during the time period under analysis, i.e. has a certain position in time (a time unit or a sequence of time units)	Rainfall, a deer at a river, a lynx chasing a deer, sunset, winter
Spatial event (spatio-temporal object)	Spatial object, event	Has certain positions in space and in time	Rainfall, a deer at a river, a lynx chasing a deer
Static spatial object	Spatial object	The spatial position in constant; exists during the whole time period under analysis	Building, village, river
Mover (moving object)	Spatial object	The spatial position changes over time	Deer, lynx, a lynx chasing a deer
Moving event	Mover, event	Exists during a sequence of time units (i.e. not instant); the spatial position changes over time	A lynx chasing a deer

A trajectory can be described by a mapping from *T*(time) to *S*(space) which is represented mathematically as

$$T \rightarrow S \tag{1}$$

for an MPO [32]. The trajectory contains the positions and the respective times when an MPO was observed. The functional mapping states that for a given time there can be at most one position in space. A trajectory is composed of a sequence of spatial events (*t*,*s*).

A spatial event represented by equation (1) consists of pairs

$$(t,s), t \in T, s \in S \tag{2}$$

where each pair defines a particular position *s* in space and a particular position *t* in time [32]. Movement events refer to elementary and composite spatial events for MPOs. In adopting these definitions, the individual spatial events associated with an encounter are known as elementary encounters, and the collection of elementary encounters between two movers are known as composite encounters [11]. A visual representation of the movement model defined previously in Table 3 is shown in Figure 5.

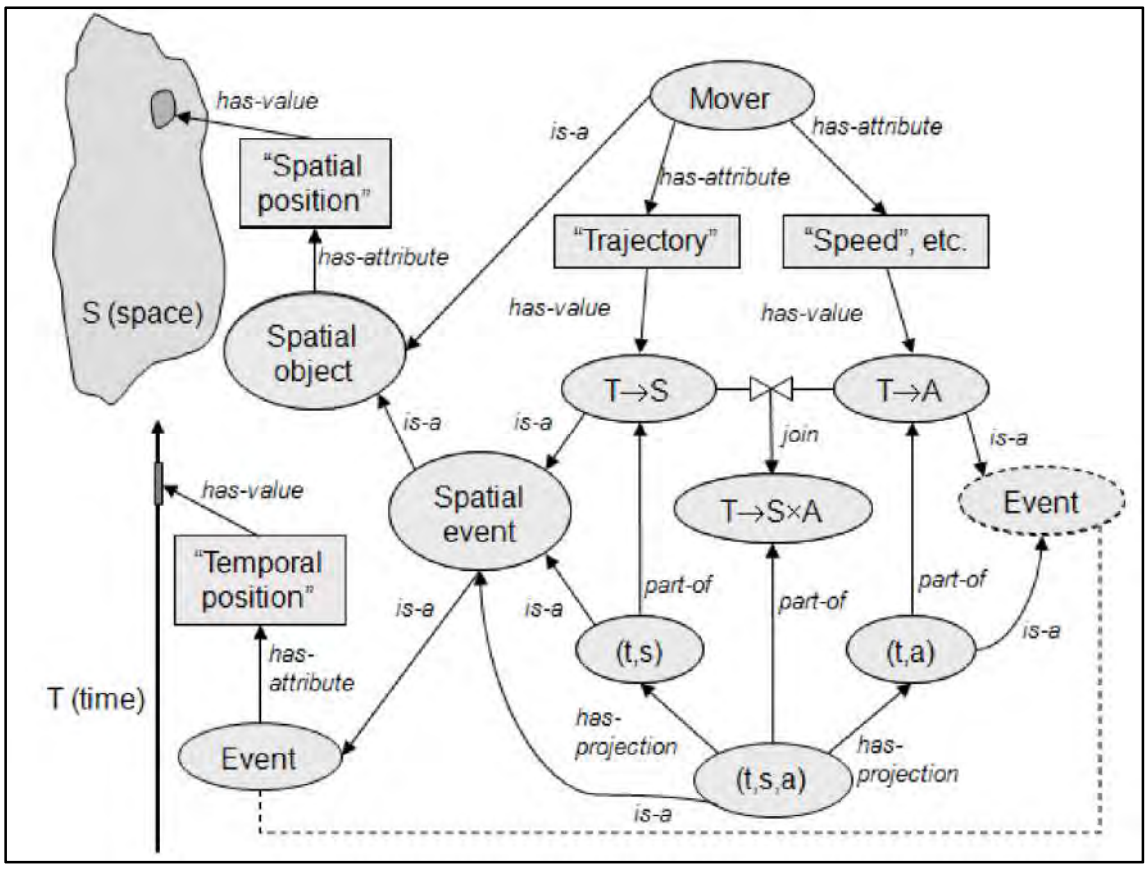

Figure 5. Visual representation of movement model from, [32].

The encounter detection process detects encounters using position records from historical AIS data. The encounter detection algorithm finds the encounter pattern using only the trajectory attributes of space *S* and time *T* shown in Figure 5. The algorithm does not require any other attributes *A* shown in Figure 5.

Movement data can be represented by the movement function

$$\mu : O \times T \rightarrow S \tag{3}$$

that emphasizes that for each object at a given time T there is at most one location S in space that it occupies [34]. The functional mapping shows that the position S is dependent upon a specific object O at a specific time T.

Movement data from a data set may be given only for a limited number of time units where the set T in

$$O \rightarrow (T \rightarrow S) \tag{4}$$

is finite and often quite small [34]. To increase the cardinality of the set T, interpolation is performed to estimate the spatial position of movers in intermediate time units between the measurements. Interpolation is allowed due to spatial and temporal dependence in movement data. Positions created using interpolation may not be valid when there are too few measured positions, and they are separated by large time intervals.

Movement data may also be divided so that only a part of the trajectory for a given mover is analyzed [34]. The movement data can be represented as

$$O \rightarrow ((T_1 \rightarrow S) \cup (T_2 \rightarrow S) \cup ... \cup (T_k \rightarrow S)) \tag{5}$$

Where T_1, T_2,…,T_k are non-overlapping subsets of the original time T in equation (4). Generally, it is not necessary that the union of T_1, T_2,…,T_k equals T. Movement data may be spliced and recombined as necessary to support processing.

When comparing the positions of movers in space, it is not meaningful to only consider two movers as being in the same place only when the coordinates are exactly the same [34]. For this reason, two positions in space should be considered as the same space if they are sufficiently close in space.

D. VISUALIZATION OF MOVEMENT

Visualizations of trajectories are difficult to comprehend for large data sets due to the large number of movers and long periods of time [34]. Prior to creating visualizations a large data set can be aggregated or filtered. Prior thesis research at NPS focused on aggregation approaches for processing large data sets [8, 9]. Aggregation of data using techniques such as clustering are a common type of data mining.

In this thesis pattern mining approaches for processing a large data set are investigated. Specifically, the use of an encounter detection algorithm for processing an archive of historical AIS data is investigated. Trajectory data is created when an encounter is detected between two movers. This trajectory data supports visualization and analysis.

To explore a large amount of data at once, an interactive dynamic data filter is implemented [11]. The filter is implemented by importing trajectories for each encounter into the V-Analytics software as a layer on a geographic map as shown in Figure 6. Additional filtering includes a spatial filter created by drawing a rectangle frame in the area of interest on the map display. A closer look at an encounter using a spatial filter is shown in Figure 7.

E. ENCOUNTER DETECTION ALGORITHM

The encounter detection algorithm arranges the position records into a set of sorted lists (SSL) so that only a minimum number of records need to be compared [30]. The algorithm performs a single sweep over the record set to arrange it into a SSL. The algorithm is embedded within an interactive visualization process as shown in Figure 8. The algorithm assumes the record set is in chronological order and requires that if the record set is not in chronological order that it be sorted at the beginning of processing.

The main encounter detection parameters that can be configured are the temporal window (ΔT) and the spatial window (ΔS) [30]. For this research, two values were selected for the temporal window ΔT and two values selected for the spatial window ΔS. These values provide a total of four different configurations for the encounter detection algorithm. The value of 30 seconds for the temporal window ΔT was chosen because it was reported in [30], and the value of 60 seconds was chosen because it was reported in [11]. The value of 0.2 nautical mile (nm) for spatial window ΔS was chosen because it was reported in [11]. The value of 0.1 nm for spatial window ΔS was chosen to provide a second value for investigating the role of spatial sensitivity in encounter detection.

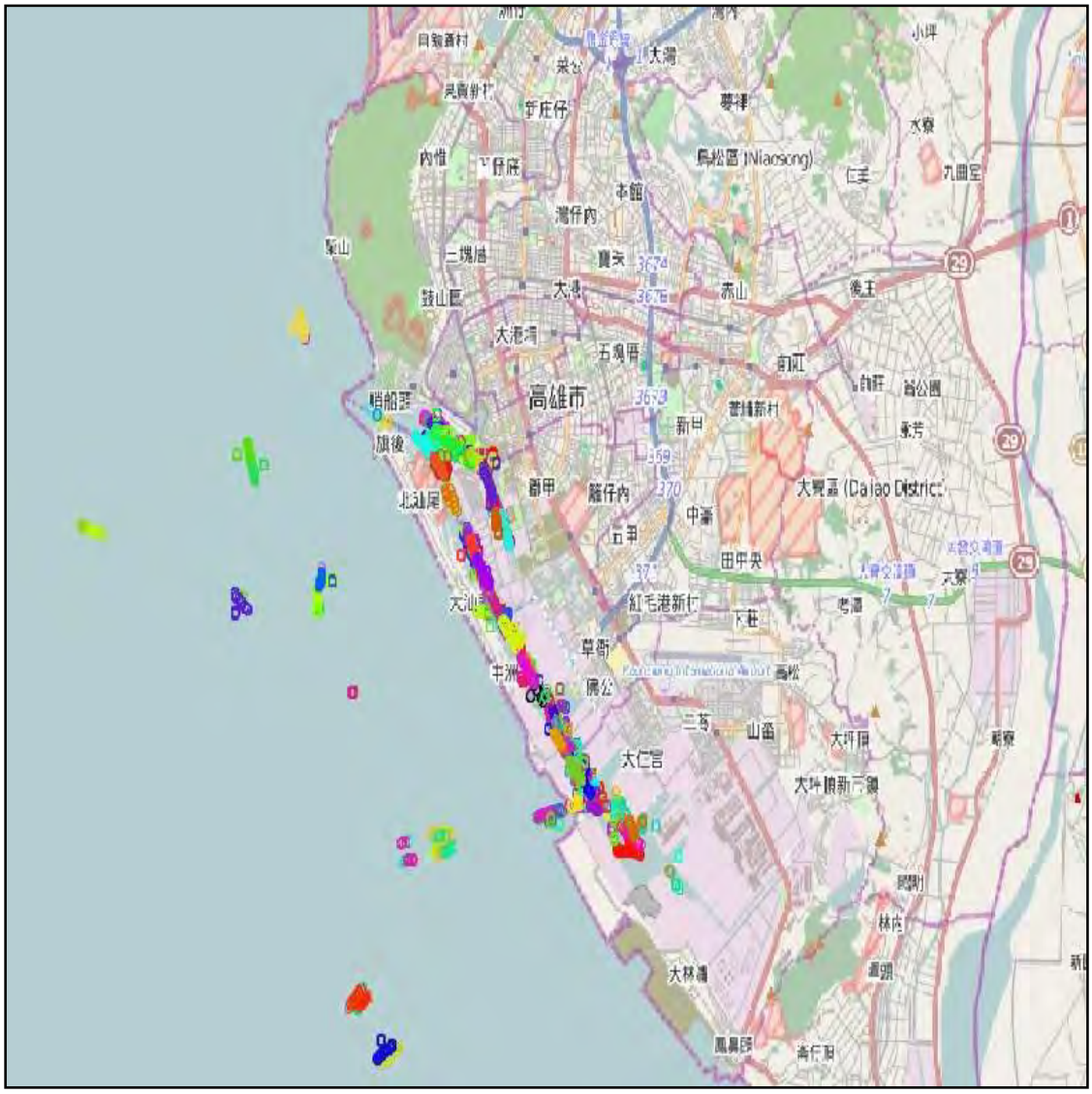

Figure 6. Geographic map showing encounters.

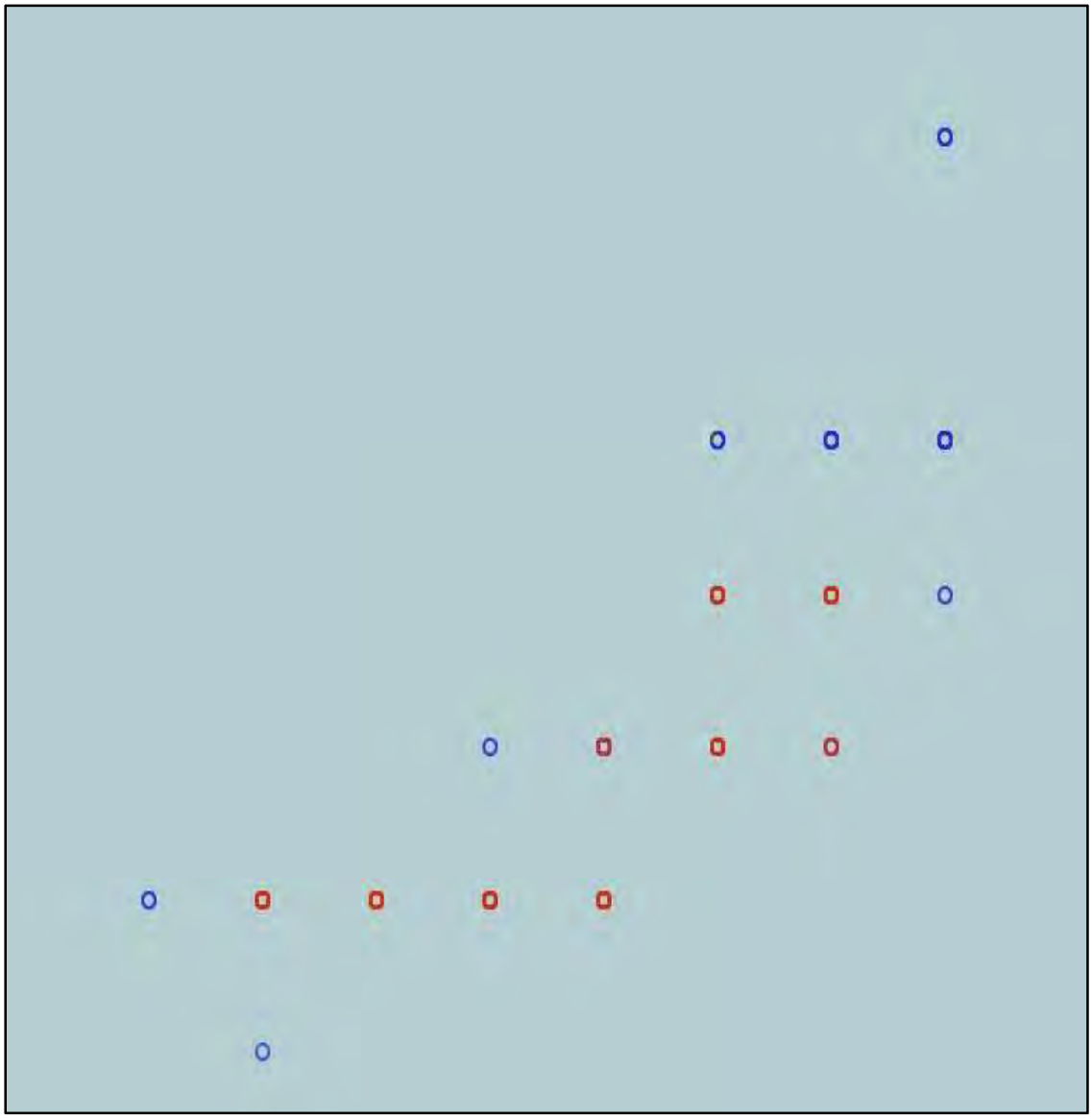

Figure 7. Close-up view of an encounter.

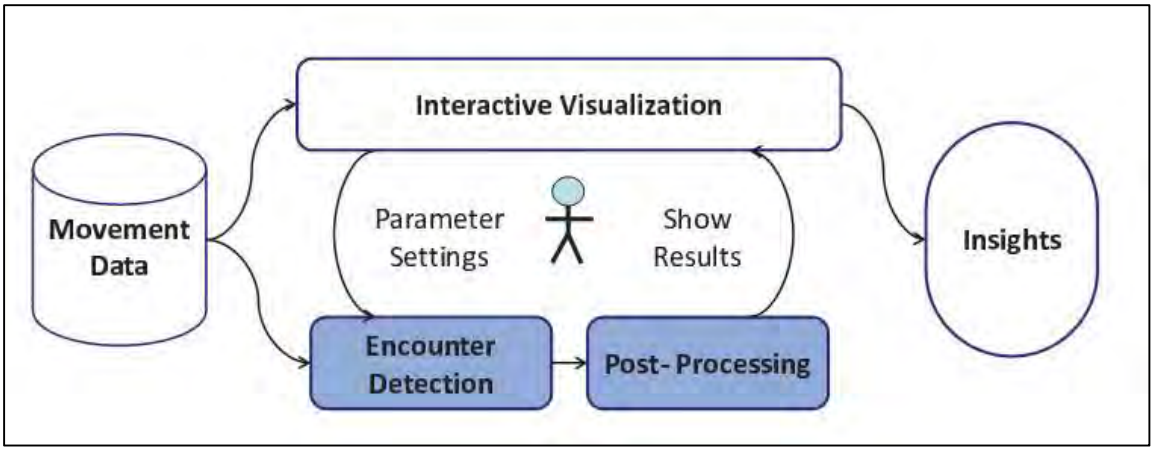

Figure 8. Interactive visualization process from, [30].

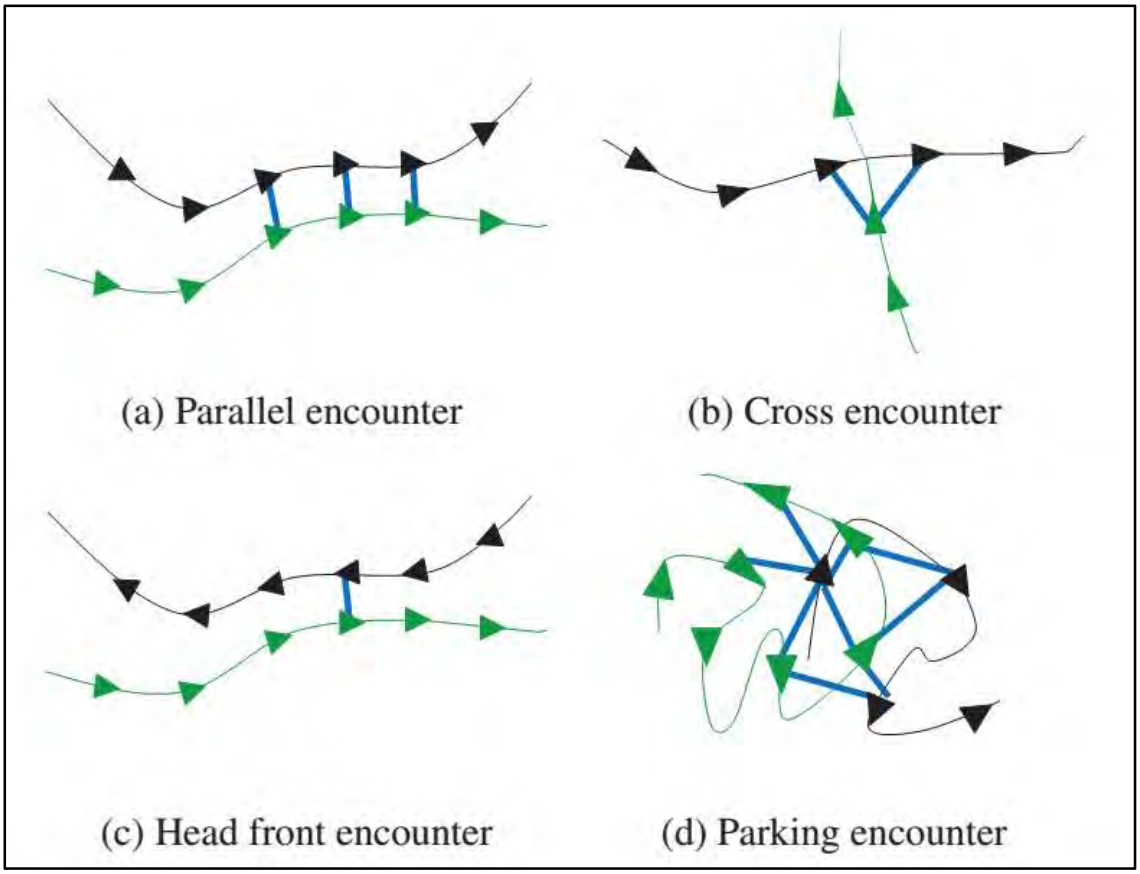

Figure 9. Encounter pattern classifications from, [30].

Characterization of encounters is required to identify entities, activities and transactions. The encounter detection algorithm does not automatically characterize encounters detected. Encounters are investigated and then characterized using a visualization in an interactive environment. Example patterns used for characterization are shown in Figure 9.

In Figure 9, each spatial event is represented by a solid triangle, where all spatial events for a trajectory of an MPO are shown in the same color (black or green). The path of the MPO is shown by connecting lines of the same color (black or green) between each of the spatial events. Encounters are represented in this example by a connecting blue line between the two MPOs involved in each encounter.

The four encounter patterns shown in Figure 9 can be identified using the velocity, the angle between the movement vectors of the MPOs and duration of the encounter [11]. For the parallel encounter pattern shown in Figure 9(a), the angle between the movement vectors should be approximately 0 degrees, and the duration of the encounter is expected to be long. For the cross-encounter pattern shown in Figure 9(b), the angle should be between 0 and 180 degrees with a short duration. The head-on encounter pattern shown in Figure 9(c) is similar, with the restriction that the angle is approximately 180 degrees. Finally, the parking encounter pattern shown in Figure 9(d), is at a low velocity, the duration of the encounter is expected to be long, and the angle of the encounter is not relevant.

There are three possibilities for specifying what an encounter algorithm finds and reports for a given record set [35]. In one case, we simply want to determine if an encounter pattern is present for a record set and report one example encounter. In a second case, we want to find and report all occurrences of the encounter pattern. For a third case, we want to report the largest size subset of encounters that meet the criteria for the encounter pattern. An encounter detection algorithm to meet the requirements of the second case that finds all occurrences of the encounter pattern is investigated in this thesis.

Recall, in the beginning of this chapter, an exact solution for the encounter algorithm was previously discussed as requiring $O(n^4)$ time. For run times specified in asymptotic notation, reducing the exponent in n^4 provides the greatest improvement when processing large record sets [35]. The asymptotic run times do not include interpolation of data points, and the algorithm being discussed only detects encounter patterns using spatial and temporal information already within the record set.

Asymptotic run times have also been derived for an approximation of the encounter pattern where the encounter pattern is defined as a set of MPOs within a range R of one another. One approach to an encounter detection algorithm is to check all possible pairs of position records in order to identify the occurrence of encounters [30]. This approach is equivalent to performing a linear search on the record set. The asymptotic run time for a linear search is $O(n^2)$ [28]. This asymptotic run time is the same as that required for a three-dimensional nearest neighbor query search [35]. Run time performance can be improved through the use of an approximate nearest neighbor query structure combined with the logarithmic method also known as a binary search method. The asymptotic run time in this case is $O(n \lg n)$. The asymptotic run times discussed here assume that the record set has sufficient spatial and temporal resolution to allow for an encounter to be detected without the need for interpolation [35].

The algorithm used in research for this thesis utilizes a SSL data structure allowing for a scalable solution to the encounter detection algorithm [30]. The SSL data structure organizes the record set for comparing each record only with records having a time difference of less than ΔT. Since the record set is chronologically ordered, only one sweep is required when executing the algorithm.

For this research, an SSL was created using a cell array data structure in MATLAB. A two-dimensional array was constructed with the first dimension defined as the rows of the array and the second dimension defined as the columns of the array. The column dimension was defined as the dimension for time with a resolution of one second since the timestamp on the AIS data logs is reported to the nearest second. The row dimension was defined to contain lists of one or more records for a given time. The data structure was created by copying each record from the AIS log to the column

corresponding to the timestamp for the record. Additional rows were added as needed to the data structure to accommodate the number of records for each list.

Each cell of the array contains a *Record* object. The *Record* object was defined using a class definition in MATLAB. Each *Record* object contains the Maritime Mobile Service Identity (MMSI), latitude, longitude and time stamp for a vessel. The time stamp is stored within the *Record* object as a day, month, year along with the hour, minute and second. The format of the *Record* object is shown in Table 4.

Table 4. *Record* class data format.

Field	Function	Class	Bytes
MMSI	ID	int32	4
Long	x position	double	8
Lat	y position	double	8
month	time	int32	4
day	time	int32	4
year	time	int32	4
hour	time	int32	4
minute	time	int32	4
second	time	int32	4

Each column of the data structure consists of a list of records with the same time value. Since the columns are in chronological sequence, it is only necessary to check within lists between the current record and ΔT previous records [30].

The algorithm processes a record list in the SSL data structure format using the following sequence of operations for each record:

1. Find the list in the SSL that contains the time for the current record.
2. Sweep over this list and previous ΔT lists and look for encounters between MPOs.
3. Add to the encounter list any encounters between MPOs with different IDs. [30]

To assess the complexity of this algorithm, the assumption is made that the records in the record list are distributed uniformly over time [30]. Consistent with the

previous discussion on the algorithm, the assumption is that there is no need for interpolation. The asymptotic time O for the complexity C of searching a list within the SSL is defined as

$$O(C(n, \Delta S)) \tag{6}$$

where n denotes the total number of records in a list within a SSL and ΔS is a specific square size due to the spatial sensitivity. The asymptotic time O for complexity C of searching the entire SSL is

$$O(NC(n, \Delta S)) \tag{7}$$

where N is the number of lists to be searched in the SSL. The asymptotic time for complexity of searching all records is

$$O(MNC(n, \Delta S)) \tag{8}$$

where M is the total number of input records to be processed. This discussion shows that the algorithm is scalable with the number of input records.

F. ENCOUNTER DETECTION VISUAL ANALYTICS PROCESS

The encounter detection visual analytics process is implemented using a combination of non-commercial software and MATLAB code developed for this thesis. The data flow for the encounter detection visual analytics process is shown in Figure 10.

The input AIS Data Log file is loaded into the AisDecoder software for data pre-processing. The AisDecoder software decodes and filters the input Log file to create Comma-separated value (CSV) output files for each of three AIS Message types. The CSV output files generated by AisDecoder contain the position records for the geographic area specified by the range filter.

Data processing begins when the CSV files are loaded into MATLAB using the *createLog* function. The *createLog* function reads in each of the CSV files and creates a single *log* object, which stores all of the position records. For each CSV file, the *createLog* function reads the file one line at a time to retrieve the position records. For each position record, the *createLog* function stores the *MMSI* identification field, the *latitude* and *longitude* fields, and the time stamp as *year*, *month*, *day*, *hour*, *minute* and *second* fields in the *log* object.

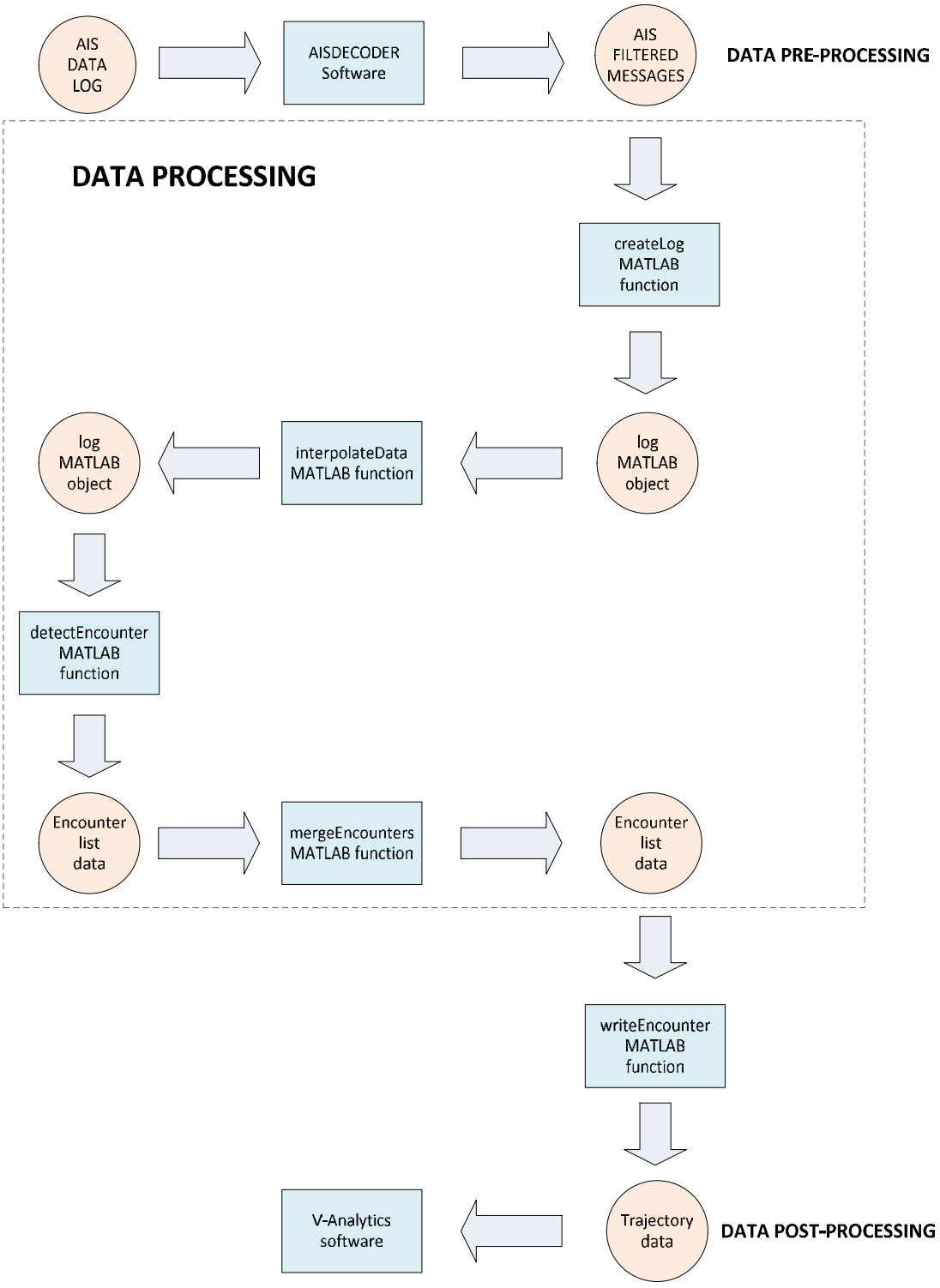

Figure 10. Encounter detection visual analytics process data flow.

The *log* object is provided to the *interpolateData* function, which performs linear interpolation on the record set. The *interpolateData* function pre-interpolates the record set in order to improve computational performance when processing large record sets and to enable the use of a spatial windowing technique described in the next section. Additional position records are added to the *log* object by the *interpolateData* function when a gap in the record set is found that exceeds the temporal window. The *interpolateData* function creates new position records by performing a linear interpolation in position and time. The *interpolateData* function provides a computationally efficient method by collecting all the records for a single ID and pre-sorting the record set by time. The recursive structure of the original encounter detection code as proposed in [30] is used to implement a recursive interpolation algorithm that when using pre-sorted records allows the interpolation to be completed in a single pass of the record set.

The *detectEncounter* function is called using the *log* object as input. The *log* object contains a time-sorted and pre-interpolated record set. The *detectEncounter* function creates the SSL using the input log object. For each list in the SSL and for each record within each list, the *detectEncounter* detects encounters that occur between records with different *MMSI* values. An encounter is detected by calculating the distance between the two position records and comparing against the value of ΔS, the spatial sensitivity. When the distance between the two positions is less than or equal to the spatial sensitivity an elementary encounter has been detected. The elementary encounter detections are then grouped into composite encounters based upon the MMSI values. Each unique pair of MMSIs involved in an encounter creates a composite encounter.

The *encounter list* is generated as an output from the *detectEncounter* function. The *encounter list* is a cell array data structure that contains all the encounter groups, where each encounter group contains all the elementary encounters for a composite encounter. Each element of a composite encounter contains, a pair of records, and the calculation of the distance between the records.

The *encounter list* created by the *detectEncounter* function is provided as input to the *mergeEncounters* function. The *mergeEncounters* function identifies duplicate

encounters from each composite encounter and deletes them. Duplicate encounters are removed from the *encounter list* when both records in an encounter pair have the same time stamp. The *mergeEncounters* function also sorts the *trajectory data* and deletes duplicate entries. The *mergeEncounbters* function creates *trajectory data* from the *encounter list* so only positions that result in an encounter detection are output. For that reason, the *trajectory data* may not be contiguous and may contain gaps in the case that two vessels are moving in and out of an encounter over a period of time.

The final MATLAB function that is called is the *writeEncounters* function. The *writeEncounters* function generates the configuration and input data files for use with the V-Analytics software. The *writeEncounters* function creates CSV format files for *trajectory data* that are loaded as tables into the V-Analytics software. The *writeEncounters* function also creates the accompanying configuration file to import the *trajectory data* files as map layers in the V-Analytics software. For each composite encounter, the *writeEncounters* function generates two CSV output files. Each CSV output file contains *trajectory data* for one vessel.

G. SLIDING SPATIAL WINDOW

During research for this thesis, it was observed that when the record set was sparse in times, the original algorithm produced extremely long computation times. The original algorithm combined encounter detection and interpolation in an integrated recursive process. For this reason, the original algorithm was modified to perform interpolation of the record set prior to encounter detection processing as suggested in [30].

Unfortunately, even with pre-interpolated data, the expected run time for processing a record set sparse in times is still extremely long. The expected run time was based on an observed run time for processing a fraction of the entire record set and extrapolating to estimate the expected run time. Even though, the sliding temporal window reduced the total number of record checks when compared to a direct comparison of every record, this approach still resulted in a very large number of encounter checks for each record.

To further reduce the number of encounter checks required of the algorithm, an additional sliding spatial window approach was considered. Investigations into this approach resulted in several prototype implementations using divide-and-conquer approaches to the encounter detection problem. The divide-and-conquer approaches follow the principles for dividing, conquering and combining as outlined in [36]. All approaches rely on a reduction in the total number of encounter checks performed by applying the sliding temporal window to only a subset of the total geographical space. The differences between the approaches are the method of dividing the geographical space and the methods to address the cases when records occur near one of the division boundaries. All approaches are verified to ensure that the algorithm always produces a correct solution and evaluated for improvement in efficiency for computation.

The first approach that was investigated consisted of dividing the record set into a series of record sets based on the values for latitude and longitude. Each record was assigned to be processed in a record set consisting of all the records within a one degree of latitude by one degree of longitude square box. For the case where a record was near the edges of the box, it was also included in the adjacent box for processing. A graphical representation of this concept is shown in Figure 11.

The circle dots in Figure 11 represent the relative position of a record within the one degree latitude by one degree longitude square box. Two examples are shown in Figure 11. The circle in the upper right corner of the solid box is included in the record set for the solid box it is contained in and, additionally, the adjacent boxes directly above, directly to the right and the upper right diagonal. All three of these boxes are designated using a dotted line border in the figure. The second example is the circle located in the middle of the bottom of the solid box. In this example, the circle is included in the record set for the solid box it is contained in and, additionally, the adjacent box directly below it. This box is also shown using a dotted line border in the figure.

In order to reduce the memory required by MATLAB, the data object only contains the current record set that is being processed. After each record set is processed, the encounter list is extracted from the data object and saved in a separate data structure. The encounter lists generated for each record set are combined during post-processing.

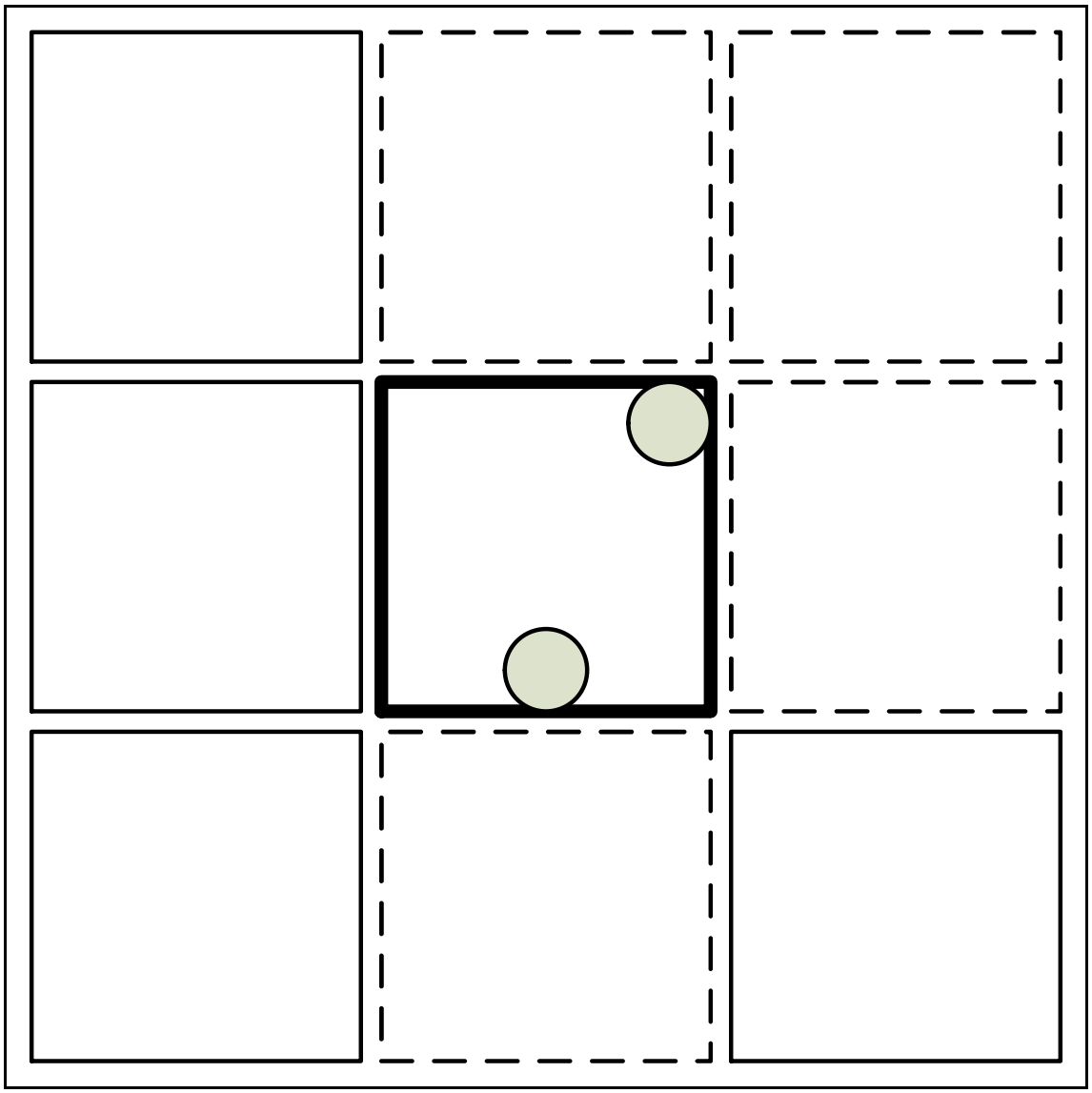

Figure 11. Graphical representation of spatial window approach.

A disadvantage of this approach is that for each square box the entire record set is checked in order to build the current record subset for processing. This approach was used to limit the amount of memory required when handling a record set with a large number of sparse times. The generation of the record subsets for this approach can require a significant amount of processing time, and for geographical areas where the

maritime traffic is dense (i.e., near ports), the reduction of encounter checks may not offset the additional processing required by the division and recombination process.

The boundary condition check was verified experimentally using a record set that could be efficiently processed without the need for the sliding spatial window approach. The boundary condition check was verified for each of the four configurations by comparing the number of encounters detected by the unmodified algorithm with the number of encounters detected by the algorithm modified to incorporate a spatial window. A summary of these results are shown in Table 5.

Table 5. Parameters and boundary conditions by configuration.

Config	N (seconds)	ΔS (m)	Boundary (degree)
1	60	185	0.001
2	60	370	0.002
3	30	185	0.001
4	30	370	0.002

As shown in Table 5 there is a direct relationship between the boundary value and the spatial sensitivity. The value of the boundary used for the first configuration is only one one-thousandth of a degree. It is important to keep the boundary value at the minimum value required in order to detect all encounters because higher values for the boundary results in additional duplicate encounters that need to be filtered out, and this requires additional time for processing.

A further extension of this concept was also investigated where the record set was first divided into a series of record subsets based on the previously discussed one degree of latitude by one degree of longitude square box and the accompanying boundary conditions. In this prototype, this division is only performed once.

Another prototype proceeds to further divide each of the record subsets into a smaller subset consisting of only the records in a smaller square box measuring a tenth of

a degree of latitude by a tenth of a degree of longitude. This prototype further reduces the processing required for the division process since the original record set is only divided once. The number of total encounter checks is also reduced significantly based on this further division of the geographical space. In comparison to the previous prototype, the geographical area has been reduced by a factor of one hundred. Further improvement in performance are realized for the cases where there are no records located within the smaller square box since it is not necessary to perform any of the algorithm functions for the case of no records.

Developing this prototype did require an additional level of complexity to incorporate the boundary cases and to ensure that no potential encounter would be missed due to the method of division used. The additional complexity was introduced due to the boundary cases and the need to search more than one of the record subsets when incorporating all the records needed for processing of each square box. Even though additional duplicate records are likely when using this prototype, these duplicates can be found and removed from the final results during post-processing.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPLEMENTATION AND RESULTS

The encounter detection algorithm and visual analytics process described in Chapter III was applied to a historical archive of AIS data. The details of the implementation of the algorithm and process are described in the first section, the computational results are summarized in the second section, and the results from data visualization techniques are presented in the third section.

A. IMPLEMENTATION

Selected data sets from a historical archive of AIS data were processed and analyzed. The implementation details for pre-processing data, encounter detection and post-processing data are described in this section.

1. Pre-processing Data

Data sets were pre-processed using AisDecoder software. Pre-processing is the first step of a KD process and performed prior to data mining. The AisDecoder software processes National Maritime Electronics Association (NMEA) and Automated Information System (AIS) messages received serially over a network or using a Log File [37]. The AisDecoder software decodes AIVDM/AIVDO sentences in the NMEA format and the included time stamp. AIVDM sentences are reports from other ships, while AIVDO sentences are reports from own ship [38]. An example AIVDM sentence is:

```
!AIVDM,1,1,,B,177KQJ5000G?tO`K>RA1wUbN0TKH,0*5C
```

where !AIVDM identifies the sentence type and the second and third fields indicate this message is complete since this sentence has one fragment and this is the first and only fragment [38]. Field four, which for this example is empty, is used for multi-sentence messages, while field five designates the radio channel code. Field five and field six are the data payload and number of fill bits required, while the *-separated suffix which, for this example, is 5C is the data-integrity checksum for the sentence. The data payload contains the AIS Messages that are decoded by AisDecoder to produce CSV output files.

The AisDecoder software decodes the message payloads and can also decode all 27 AIS Message types shown in Table 6.

Table 6. AIS message types, from [38].

Message 01	Position Report Class A
Message 02	Position Report Class A (Assigned schedule)
Message 03	Position Report Class A (Response to interrogation)
Message 04	Base Station Report
Message 05	Static and Voyage Related Data
Message 06	Binary Addressed Message
Message 07	Binary Acknowledge
Message 08	Binary Broadcast Message
Message 09	Standard SAR Aircraft Position Report
Message 10	UTC and Date Inquiry
Message 11	UTC and Date Response
Message 12	Addressed Safety Related Message
Message 13	Safety Related Acknowledgement
Message 14	Safety Related Broadcast Message
Message 15	Interrogation
Message 16	Assignment Mode Command
Message 17	DGNSS Binary Broadcast Message
Message 18	Standard Class B CS Position Report
Message 19	Extended Class B Equipment Position Report
Message 20	Data Link Management
Message 21	Aid-to-Navigation Report
Message 22	Channel Management
Message 23	Group Assignment Command
Message 24	Static Data Report
Message 25	Single Slot Binary Message,
Message 26	Multiple Slot Binary Message With Communications State
Message 27	Position Report For Long-Range Applications

AisDecoder was used to decode input Log Files and create CSV output files for AIS Message types 1, 2 and 3. AIS Message types 1, 2 and 3 are position reports and are typically broadcast every 2 to 10 seconds while vessels are under way and every 3 minutes while the vessel is anchored [38]. Input filtering by AIS Message type was used to generate a separate CSV output file for each message type.

Range filtering was also used to select a geographical region as specified by latitude and longitude values. A geographic area within the Tri-Border Area (TBA) of Southeast Asia, which includes Malaysia, Indonesia and the Philippines, was selected by specifying the minimum and maximum values for longitude and latitude. A configuration for AisDecoder was generated for each of the AIS Message types. The *Options* settings for decoding and filtering type 1 messages are shown in Figure 12.

As can be seen in Figure 12, AisDecoder is configured to generate output files containing records with longitude between 90 and 125 degrees and latitude between -10 and 8 degrees. This geographic region was chosen because it contains Singapore, which is a busy port, while excluding the maritime areas outside of the TBA. Including a major port such as Singapore in the maritime area under investigation required processing data sets with a large number of position records.

2. Processing Data

Encounter detection processing was implemented in MATLAB using a script to execute a main loop. The loop is executed once for each of four defined processing configurations. The processing configurations are defined by the values for the spatial (N) and temporal (ΔS) sensitivities as shown in Table 5 in the previous chapter. The *configSetup* function is called at the start of each loop to configure N and ΔS. During the first iteration of the loop, the *createLog* function is called which reads the input Log file and creates the *log* object. On the first and third iterations of the loop, the *interpolateData* function is called. The first time the *interpolateData* function is called is during the first iteration of the loop, and the record set in the *log* object is interpolated using a value of 60 seconds for N. The second time the *interpolateData* function is called is during the third iteration of the loop, and the record set in the *log* object is interpolated using a value

of 30 seconds for N. This approach provides computational efficiency by reusing the *log* object and only performing interpolation once for each value of N. The *detectEncounter* function is called during each of the four iterations of the loop. The values for N and ΔS are provided to the *detectEncounter* function as inputs. For each time through the loop, a new data structure is created for *encounter lists*. Likewise, unique *trajectory data* is created following calls to the *mergeEncounters* and *writeEncounters* functions. Encounter detection and computational statistics are generated and reported in the MATLAB status window at the end of the loop.

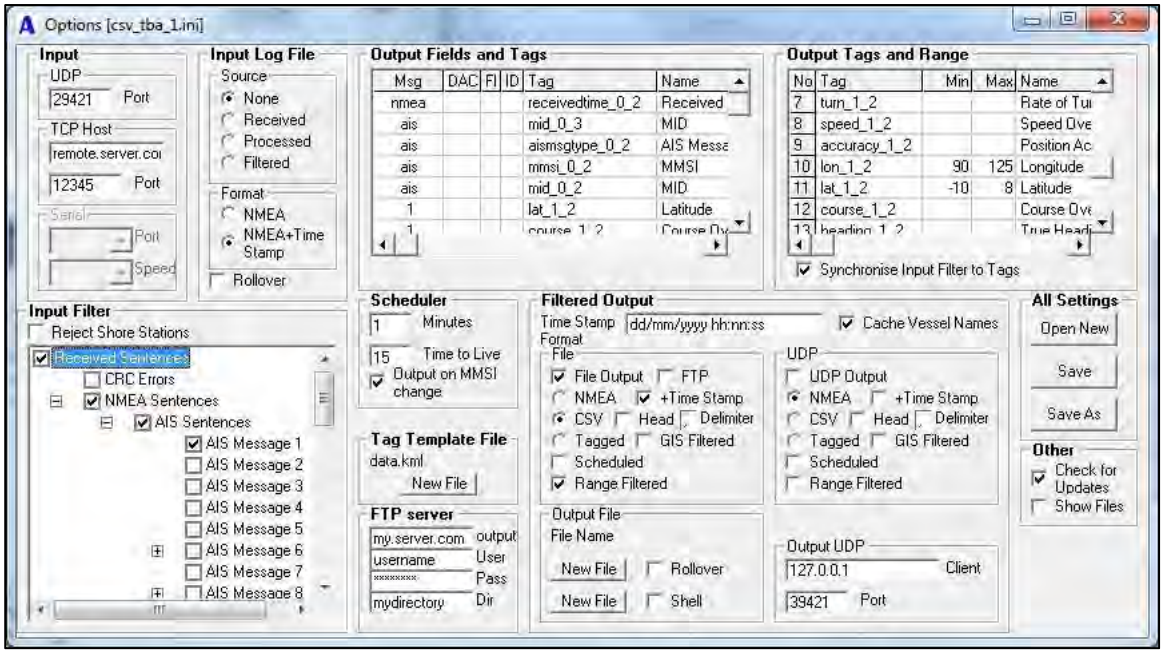

Figure 12. Aisdecoder settings window

The *createLog* function creates the initial *log* object during the first iteration of the main loop. For each of the three AIS Message types, the *createLog* function calls the *initialize* method of the *Log* class. The *initialize* method reads new records from the input CSV file line by line using a *while* loop until the end of the file is reached. For each line of the input CSV file, the *initialize* method extracts the date and time and stores the *year*, *month*, *day*, *hour*, *minute* and *second* in the respective fields in a position record within the *log* object. The *initialize* method additionally parses each line to record the MMSI,

latitude and longitude into the *MMSI*, *Lat* and *Long* fields in the position record. The *initialize* method reports the current configuration and elapsed time to the MATLAB display window every time 1000 records have been read.

The *interpolateData* function creates additional position records using recursive calls to the *detect* method of the *AISData* class. The *interpolateData* function processes a record set by using a *for* loop. The beginning of the loop creates an *sdata* object using the *AISData* class definition. The *sdata* object is used later during recursive calls to the *detect* method. During the loop, we check each record against a list of MMSIs already processed. If the *MMSI* has not already been processed, then a list containing all of the records that match the *MMSI* is created. This list is then time sorted using an insertion sort algorithm. The time-sorted list is used for another *for* loop that calls the *detect* method once for each record in the list. The *detect* method creates new position records when interpolation is required. At the beginning of the *detect* method, the current record is added to the *sdata* object. The *interpolateData* function reports the current configuration and elapsed time to the MATLAB display window every time 10,000 records have been read for interpolation.

The *detect* method searches forward and backward through the position records in the *sdata* object to find other position records with the same *MMSI*. When a position record is found searching backward, the *found_prev_flag* is set, and when a position record is found searching forward, the *found_next_flag* is set. For the case of the *found_prev_flag* being set, the time of the previous record is subtracted from the time of the current record to determine the time difference. The calculated time difference is compared with the value for N, and if the time difference is greater than N, a linear interpolation is performed. A similar calculation for the time difference is used for the case of the *found_next_flag* being set. Likewise, when the time difference is greater than N, a linear interpolation is performed. In either case, the *detect* method is called recursively using the new record as the input. The linear interpolation is performed by calculating the midpoint between the two records in both time and space. For simplicity, the midpoint in space is calculated simply by calculating the midpoint separately in both latitude and longitude.

The *detectEncounter* function detects encounters between position records with different MMSIs. For the remainder of this section, the positions records are referred to simply as records. The *detectEncounter* function uses a *for* loop to build the SSL from the time sorted pre-interpolated data in the *log* object. The *detectEncounter* function also uses a *for* loop to check for and remove duplicate records within the SSL. Encounter detection is performed by the *detectEncounter* function using nested *for* loops to check each record in the list for all of the lists in the SSL. Within the inner loop, the *detectEncounter* function calls the *edetect* method of the *AISData* class.

The *edetect* method is called once for each of the records in the SSL. The *edetect* method searches the lists within the SSL using a sliding time window. Since the SSL is pre-interpolated and time sorted, the upper edge of the sliding window is simply set to the time of the current record being checked. The lower edge of the sliding window is calculated by subtracting the value for N from the time of the current record being checked. In the case where a negative time would result, the lower edge of the sliding window is set to the first record of the SSL. The *edetect* method uses nested *for* loops to check the current record against all records in the current list and all other lists in the SSL that are within the sliding time window. Within the inner loop, a comparison is made between the MMSI of the current record and the record being checked. If the MMSI values are different, the two records are checked for an encounter.

An encounter is detected when the distance between the two records is calculated to be less than the value of ΔS. The distance between the records is calculated as the distance between the two vessels using the *distance* function, which is a function available in the MATLAB Mapping Toolbox. The *distance* function uses the latitude and longitude values for each of the records as input. The *distance* function also requires the *meanradius* to be input. The value of *meanradius* was set to the mean radius of the earth with a value of 6371.009 kilometers.

If a previous encounter pair can be found in an existing composite encounter, then the *detectEncounter* method adds the new encounter to an existing composite encounter. The composite encounters are also simply referred to as encounter groups. If the encounter pair cannot be found in an existing group, then a new encounter group is

created. The *detectEncounters* method reports the current configuration and elapsed time to the MATLAB display window every time 1000 records have been checked for encounters.

The *mergeEncounters* function identifies and removes duplicate encounters from the *encounter lists* generated by the *detectEncounter* function. The *mergeEncounters* function uses nested *for* loops to check each encounter within each encounter group for duplicates. Within the inner loop, the *mergeEncounters* function compares the *MMSI*, *Long*, and *Lat* fields and the calculated time (from the *hour*, *minute,* and *second* fields) between two records. If a match is found, the *mergeEncounters* function deletes the duplicate record.

The *mergeEncounters* function also creates the *trajectory data* by removing duplicate time entries from the *encounter lists*. Two nested *for* loops are used to find duplicate entries in both the first and second lists created by the set of encounter pairs within each encounter group. Recall that each encounter group consists of a list of encounter pairs, where each pair in the group contains two lists of records with the first list corresponding to the first vessel as determined by the MMSI and the second list corresponding to the second vessel as determined by the MMSI. To create the *trajectory data,* the *mergeEncounters* function identifies duplicates within each of the two MMSI lists and removes them. The *trajectory data* no longer maintains the relationship of encounter pairs but is a reduced data set where each list is a time ordered list of records for when the encounter condition was met. The data reduction performed by the *mergeEncounters* function reduces the number of records that require post-processing. The *mergeEncounters* function reports to the MATLAB display window the number of entries found and the number of entries deleted as each encounter group is processed.

3. Post-processing Data

The *writeEncounters* function is the final MATLAB function called by the main loop. The *writeEncounters* function prepares the *trajectory data* for reading into the V-Analytics software. The *writeEncounters* function creates the *.app* project description file for loading a project into the V-Analytics software. The *writeEncounters* function also

creates CSV output files using the *trajectory data*. Each CSV output file contains the trajectory for one vessel. Each CSV output file is loaded into the V-Analytics software as a layer on the map.

The V-Analytics software is a Visual Analytics System for Spatial and Temporal Data developed for performing exploratory analysis of spatial and temporal data [39]. Once a project is loaded into V-Analytics, the map is zoomed in on the region that contains encounters. An example data set is shown in Figure 13. A view of the same data set after zooming out once is shown in Figure 14. In this example, all the detected encounter activity is near the port of Singapore.

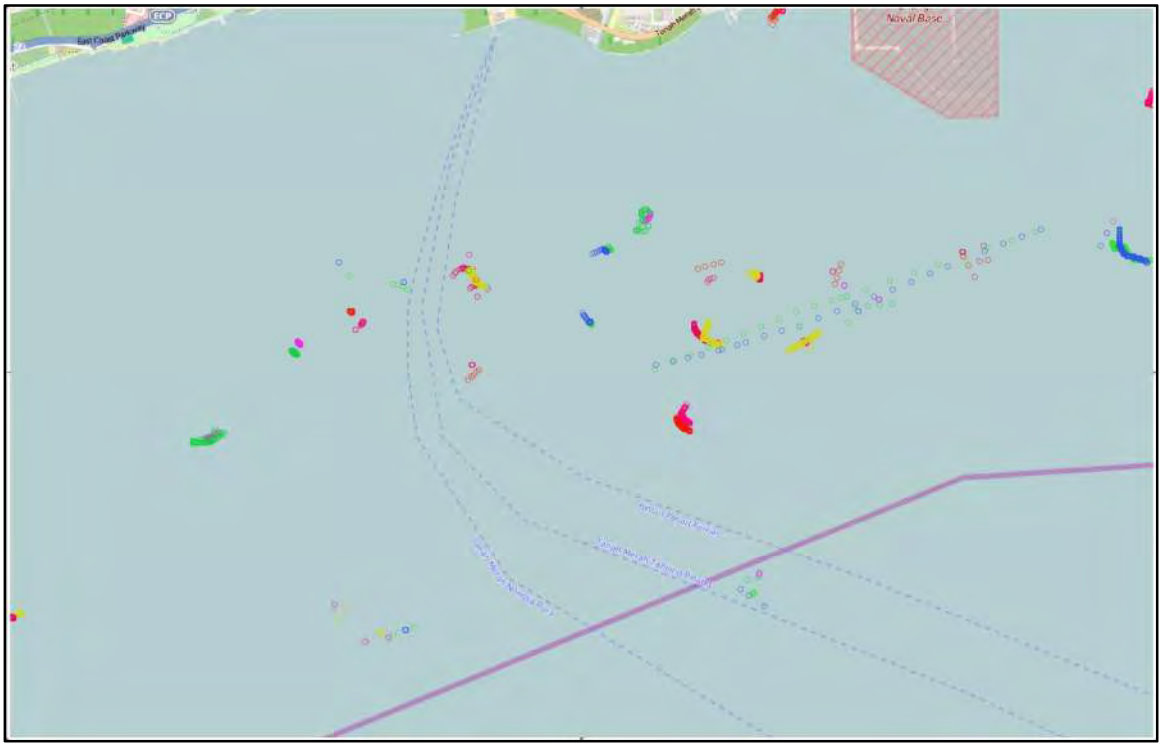

Figure 13. Default view of map showing encounters.

Figure 14. Zoomed out view of map showing encounters.

B. COMPUTATIONAL RESULTS

Selected data sets from a historical archive of AIS data were processed and analyzed. The computational results for pre-processing data and encounter detection are described in this section.

1. Pre-processing Data

The AisDecoder software was used to geographically filter AIS messages stored in Log Files. Archives of Log Files from February 2011 and January 2012 were used. The numbers of messages extracted from each data set are shown in Figures 15 and 16. As expected, the typical number of messages extracted from each data set increased from 2011 to 2013. In both the 2011 and 2012 data sets, at least one data set contained a very large number of records. The 2013 data set contains no sets with a very large number of records, and the overall distribution appears to be more even than the 2011 and 2012 data sets.

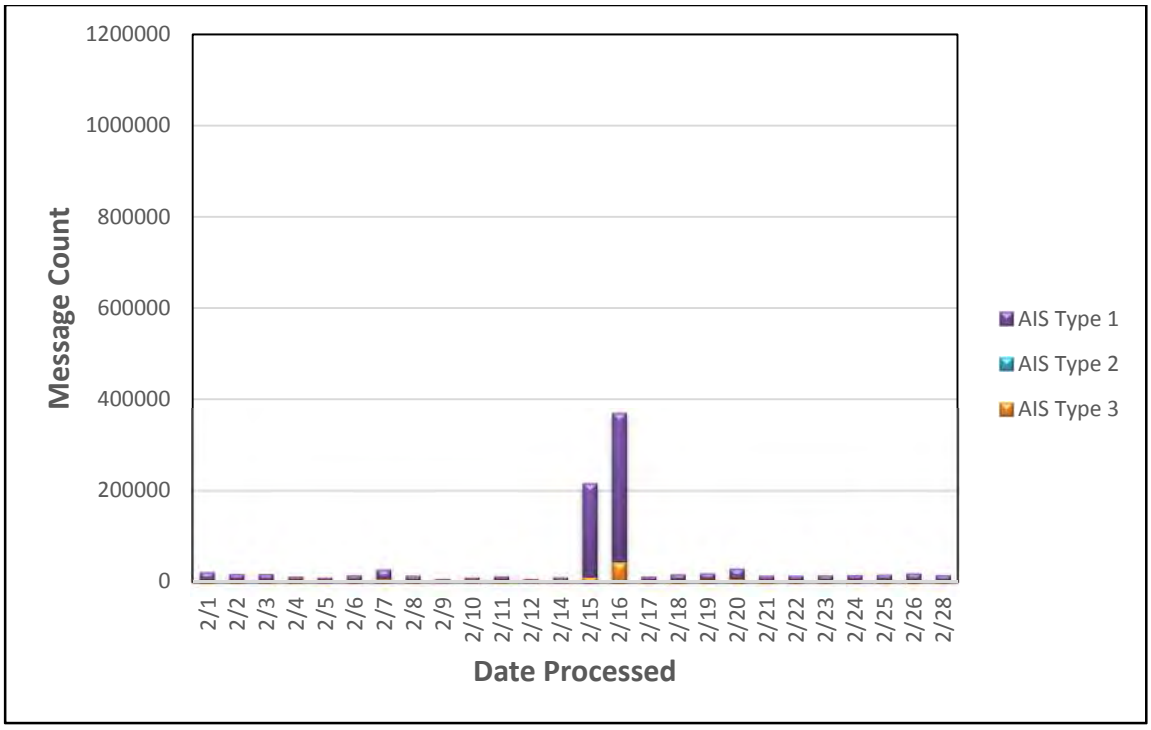

Figure 15. AIS messages processed for February 2011.

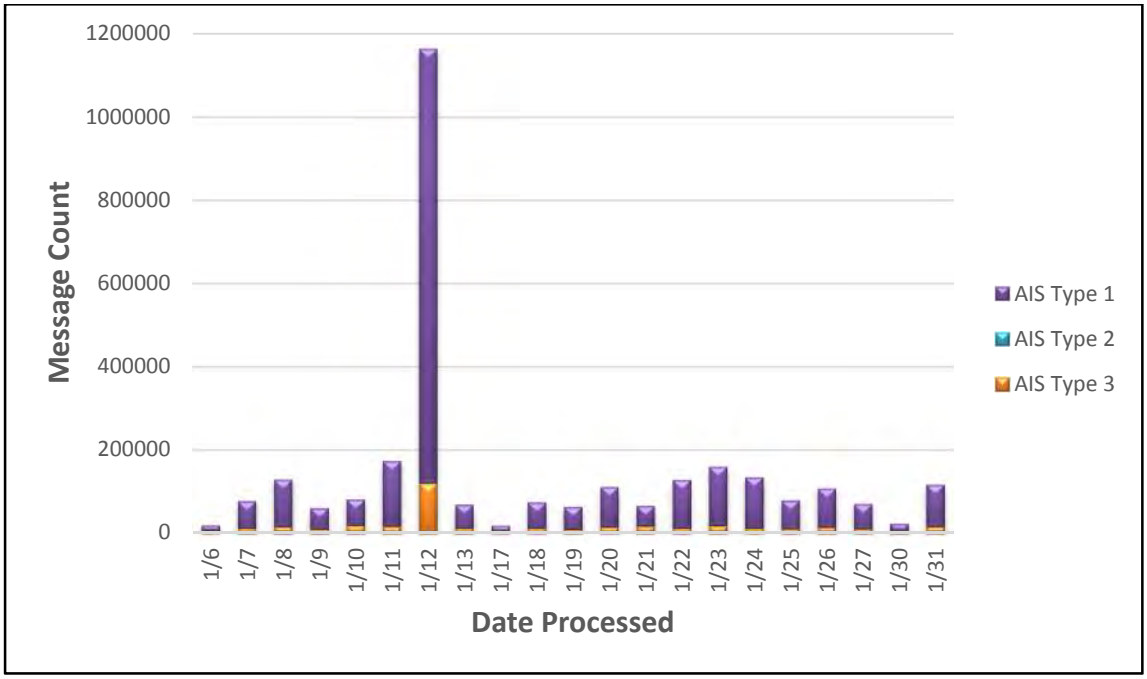

Figure 16. AIS messages processed for January 2012.

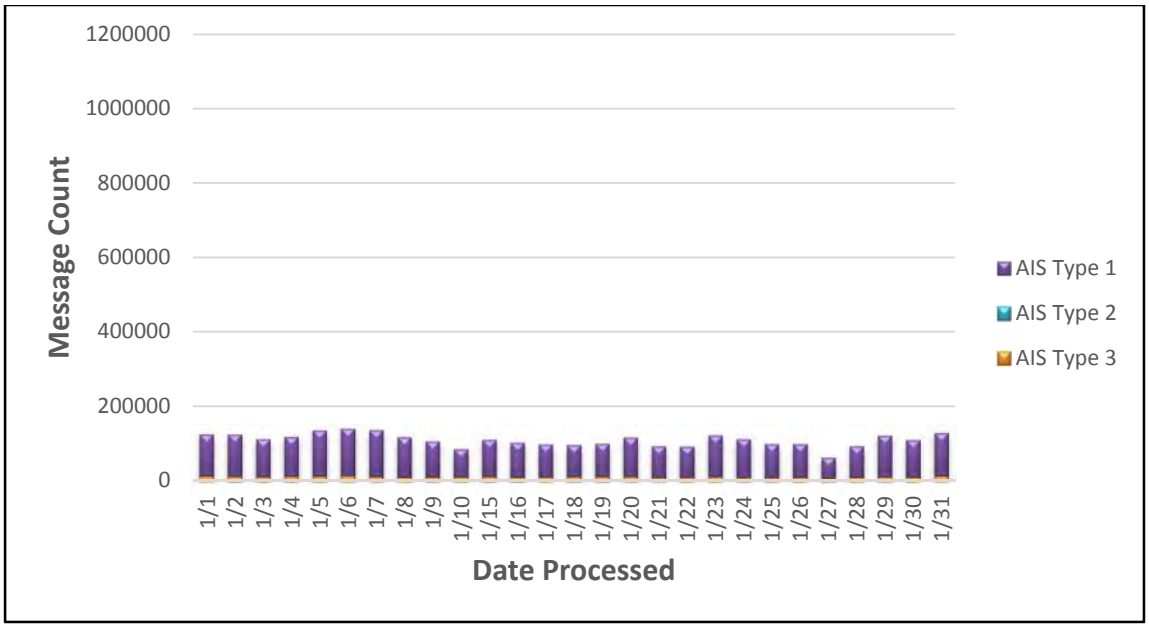

Figure 17. AIS messages processed for January 2013.

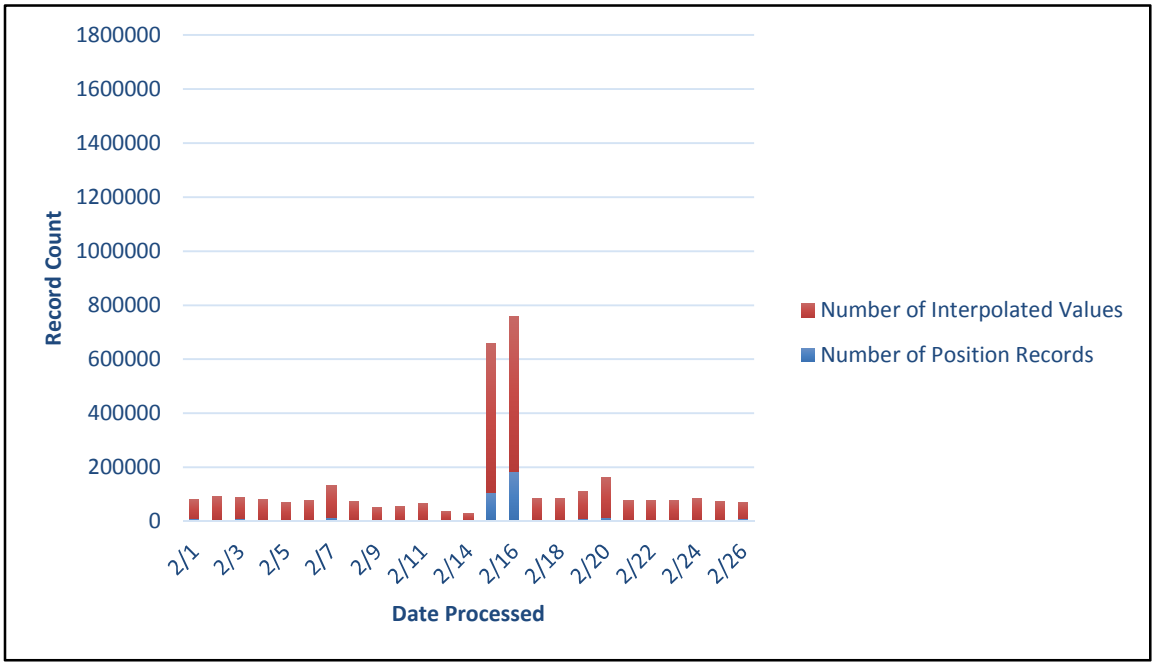

Figure 18. Interpolation processing for February 2011 for N = 60.

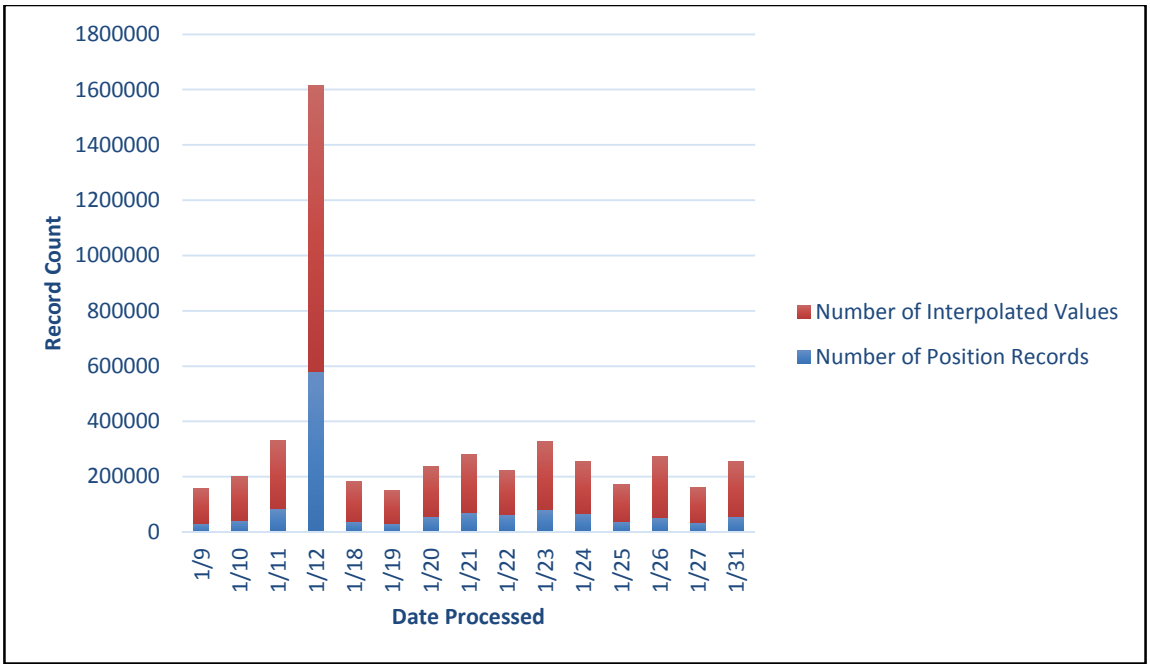

Figure 19. Interpolation processing for January 2012 for N = 60.

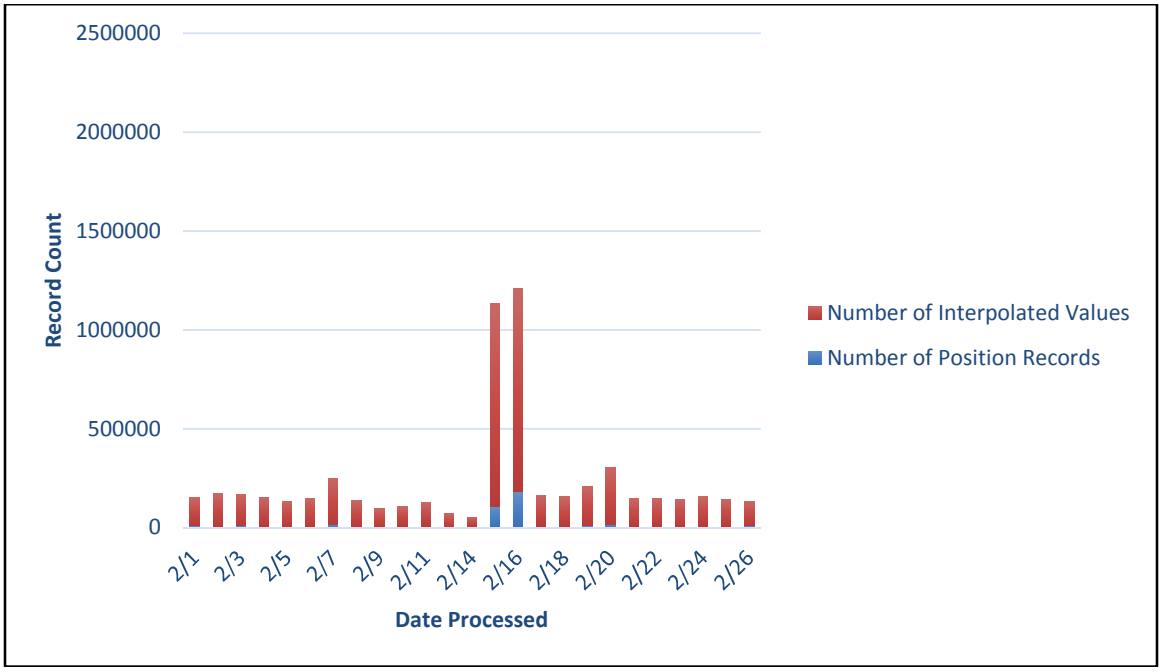

Figure 20. Interpolation processing for February 2011 for N = 30.

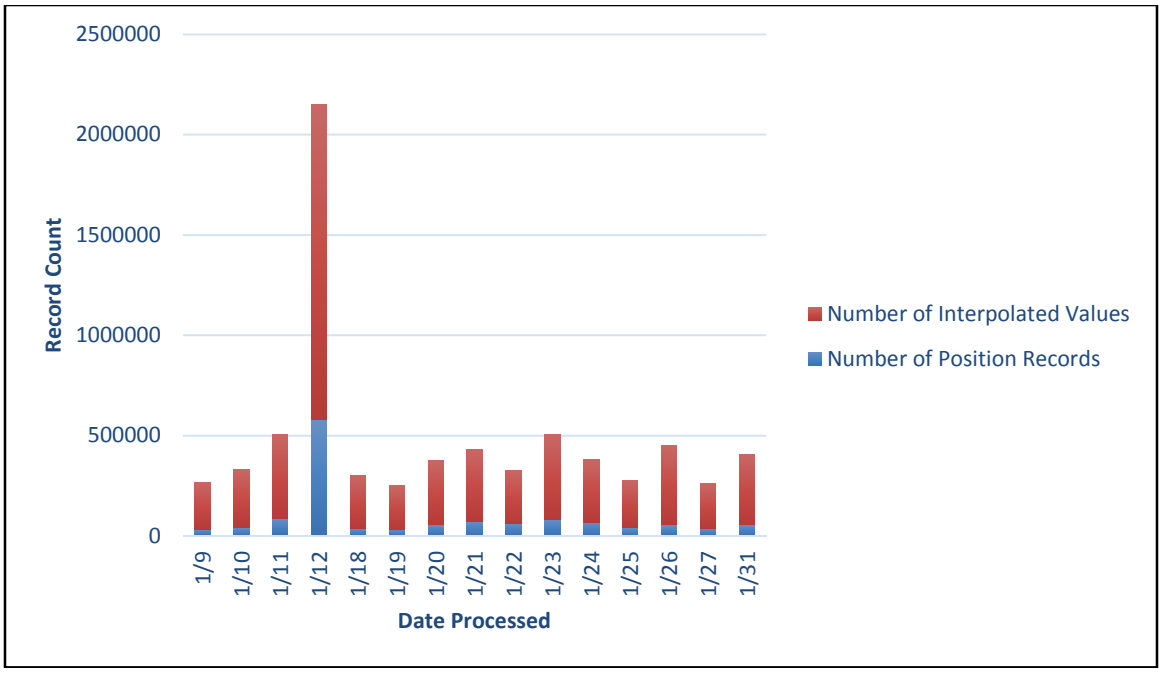

Figure 21. Interpolation processing for January 2012 for N = 30.

2. Processing Data

Interpolation is performed on the data set prior to invoking the encounter detection algorithm. Following the interpolation process, the original records and the additional records created using interpolation are processed by the encounter detection algorithm. The result of this process is summarized in Figures 18 and 19 for a value of N = 60 and Figures 20 and 21 for a value of N = 30. The archive from 2011 contains fewer records but requires more interpolation than the archive from 2012. The archive from 2013 was not processed because the larger number of messages was unable to be processed due to time constraints. Suggestions for further research are discussed in the next chapter.

The encounter detection results are summarized in Figures 22 and 23. The number of encounters generally increases for each successive configuration as defined for Config 1 through Config 4. It is clear that the Config 1 definition of N = 60 seconds and ΔS = 0.1 nm results in the least number of encounters for any data set. It is also clear that the Config 4 definition of N = 30 seconds and ΔS = 0.2 nm results in the most number of encounters for any data set.

It is an interesting result that when comparing results from Config 2 and Config 3 that either may detect more encounters than the other, and the result is dependent upon the data set that is processed. Recall that Config 2 doubles the spatial sensitivity by increasing ΔS to 0.2 nm, while Config 3 doubles the temporal sensitivity by decreasing the temporal window from N = 60 seconds to N = 30 seconds. The encounter detection results as shown in the figures show no clear ordering between Config 2 and Config 3.

An anomaly in the results can be seen when processing the very large February 15, 2011, February 16, 2011 and January 12, 2012 data sets. For these data sets the number of encounters for Config 2 is clearly greater than the Config 3. These large data sets are explored further in the next section on visualization results.

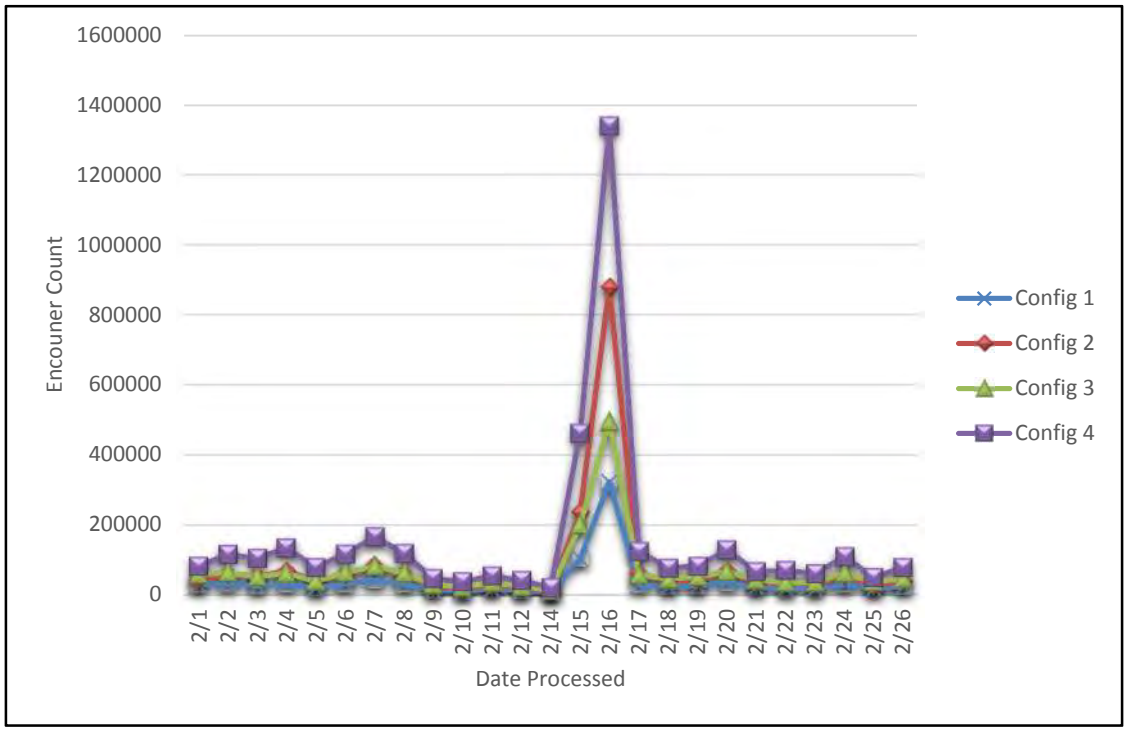

Figure 22. Encounter detection processing for February 2011.

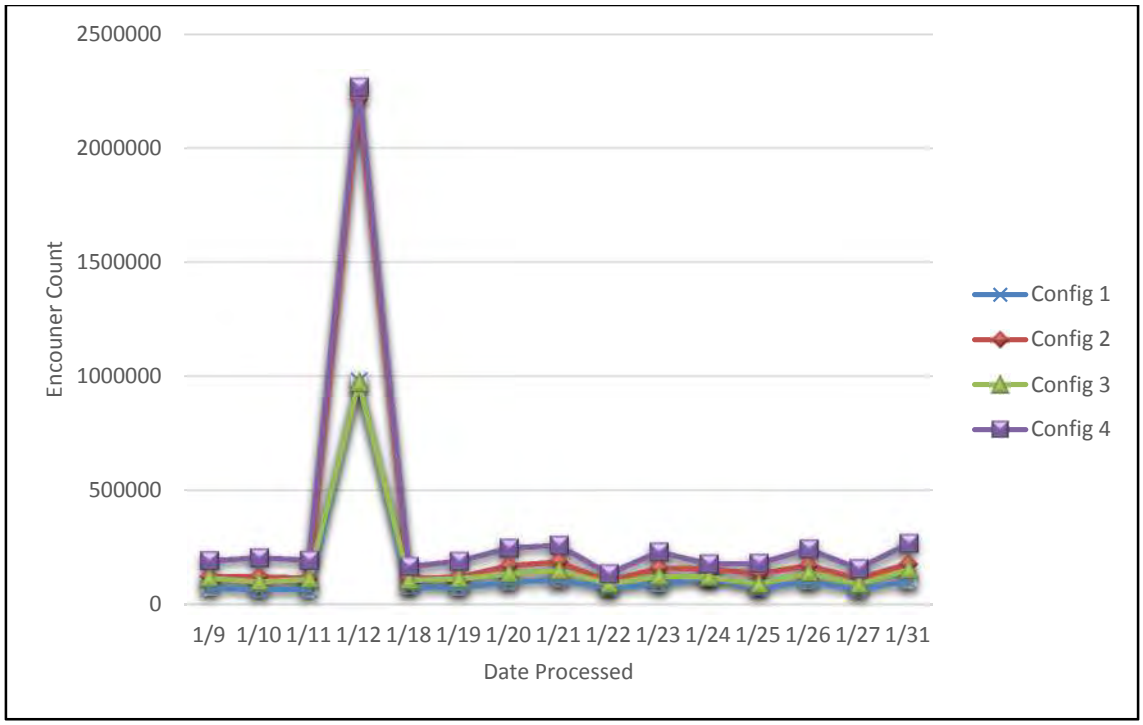

Figure 23. Encounter detection processing for January 2012.

C. VISUALIZATION RESULTS

Data visualizations created from a historical AIS archive are described in this section. A criteria for anomalous behavior is defined and the difference between visualizations created using the same data set is compared when varying the configuration parameters in Subsection 1. Examples of anomalous behavior that can be observed in the visualizations are investigated and examples of the types of patterns that can be observed are investigated in Subsection 2. The results presented here are summarized in Subsection 3.

1. Analysis of Visualizations

Visualizations were generated using each of the four configuration settings for each data set. Each visualization was reviewed to determine if any anomalous behavior can be observed. Recall that earlier in this chapter, it was discussed that when a project is loaded into the V-Analytics software, the map is automatically zoomed in on the region that contains encounters. This feature is utilized to quickly determine for each

visualization whether or not anomalous encounters can be observed. Anomalous behavior can be seen when encounters are observed apart from expected locations.

Recall that, as discussed previously, a range filter was applied to the unprocessed AIS data during pre-processing using the AisDecoder software. Within this geographic area, the main area of activity is the maritime region near Singapore. For the case when the map is zoomed into the maritime region near Singapore, no anomalous activity can be seen. For the case when the map is zoomed out to include additional area, encounters are seen within that may represent anomalous activity. Of particular interest are encounters that occur in open sea areas away from typical sea routes. Visualizations for example cases with no visible anomalous activity are shown in Figures 24 – 27.

When comparing Figure 24 and Figure 25, it can clearly be seen that there are more encounters in Figure 25. With a closer look it can be seen that the zoom-in location for Figure 25 is offset slightly to the north when compared with Figure 24. The reason for this is seen in the top of both figures. The geographic area with encounters to the northeast of the port is clearly larger in Figure 25 when compared with Figure 24. Recall thatConfig 2 used spatial sensitivity twice that of Config 1. While both Config 1 and Config 2 used the same temporal sensitivity, it can be seen that, as expected, the spatial sensitivity has affected visualizations generated.

When comparing Figure 24 and Figure 26, we see that these figures share the same zoom-in, and the locations of the encounters appear to be the same. Figure 26 was created using Config 3, which has the same spatial sensitivity as Config 1 but uses a value for temporal sensitivity that is one half that used for Config 1. The same similarity is seen when comparing Figure 25 and Figure 27. These figures use the same spatial sensitivity, and Figure 27 created using Config 4 has a temporal sensitivity that is one half that used for Config 2.

Figure 24. Visualization of encounters in Singapore maritime area (Config 1).

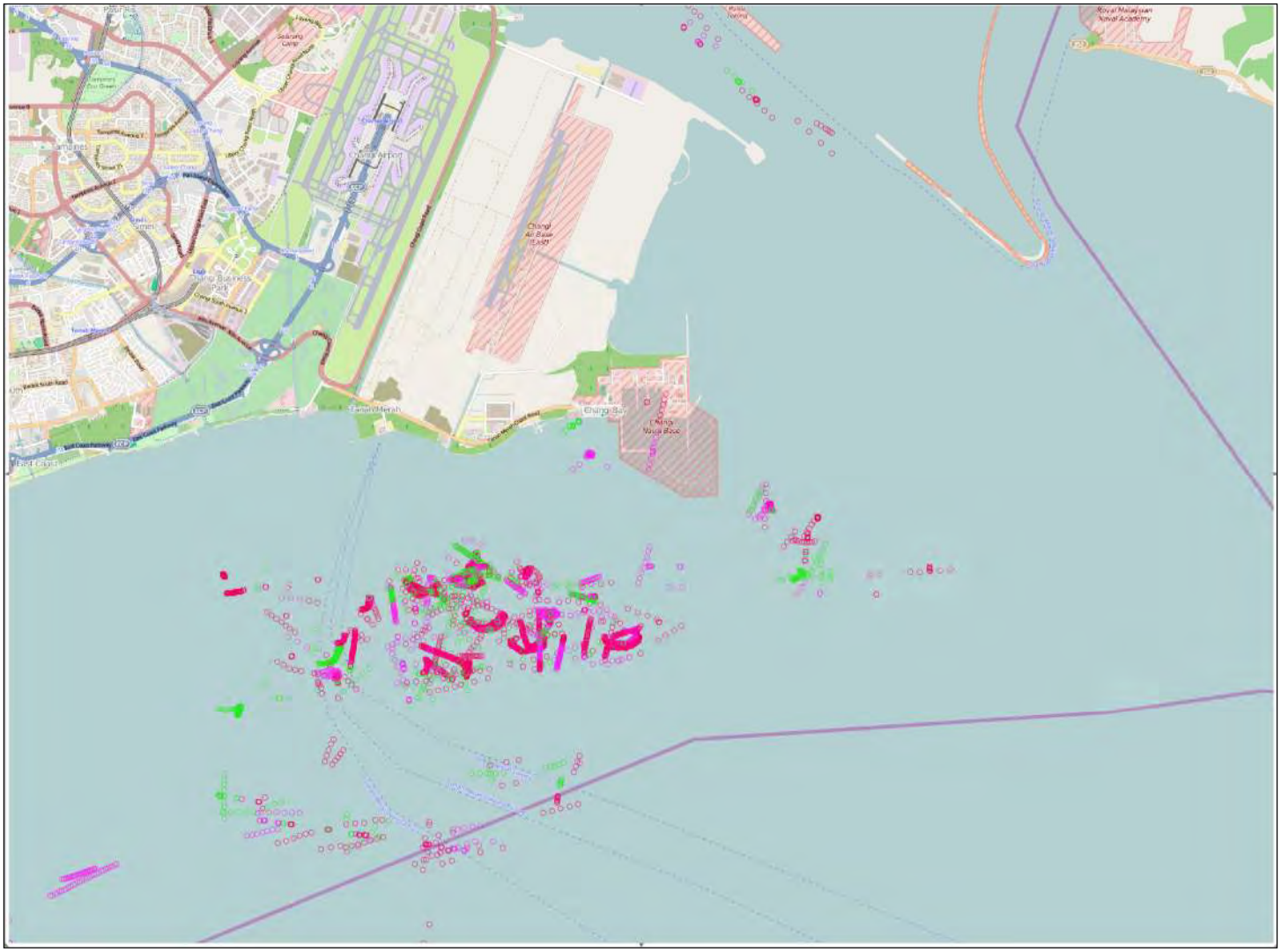

Figure 25. Visualization of encounters in Singapore maritime area (Config 2).

Figure 26. Visualization of encounters in Singapore maritime area (Config 3).

Figure 27. Visualization of encounters in Singapore maritime area (Config 4).

All of the data sets from February 2011 were processed using each of the four processing configurations and analyzed for anomalous behavior. No clear anomalous behavior can be seen in these data sets; however, the previously mentioned very large data sets did produce more interesting visualizations. Since it has already been established that there is a negligible difference between the visualizations for Config 1 and Config 3 and between Config 2 and Config 4, only the visualizations for Config 1 and Config 2 are compared.

While it is possible that some of the isolated encounters may represent anomalous activity, these findings are suspect due to the unusual size of the input data sets. In order to characterize normal behavior and detect anomalous activity, the characteristics of the data sets need to be similar to support patterns of life analysis and likewise for generating methods to detect anomalies. Example visualizations from the very large data sets discussed here are shown in the accompanying figures. Figure 28 and Figure 29 are created from data sets from February 15, 2011. Figure 30 and Figure 31 are created using data sets from February 16, 2011.

2. Anomaly Investigations

Examples of potential anomaly investigations using the AIS archive from January 2012 are provided in this subsection. Two data sets containing potentially anomalous encounters are investigated here. The two data sets are from January 10, 2012 and January 11, 2012. For the January 10, 2012 data set, two potential anomalous encounter areas are identified north of the Singapore maritime area in the open sea. A closer look at each of these encounter areas was performed by zooming into the area on the map.

The first encounter area indicates a parallel encounter pattern or a head front pattern and is shown in Figure 32. Recall encounter patterns were previously defined in Figure 9. Each vessel involved in the encounter is shown in the figure with a black box around the trajectory points. The vessel on the left is shown in a purple color, while the vessel on the right is shown in a red color. In this visualization the direction of movement is not shown so the vessels may have been traveling in parallel for a short time or, more likely, passed one another going opposite directions.

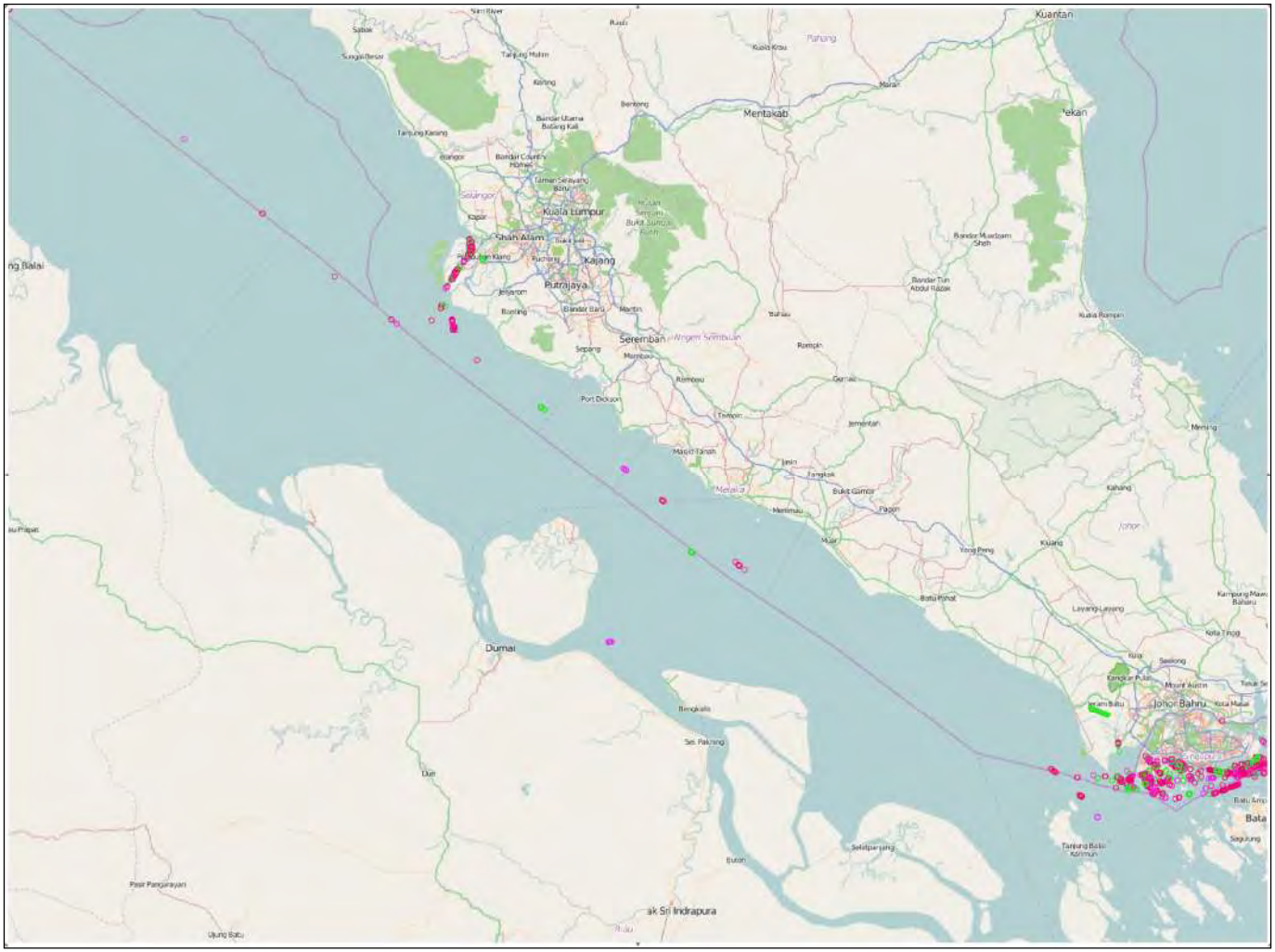

Figure 28. Visualization of encounters February 15, 2011 (Config 1).

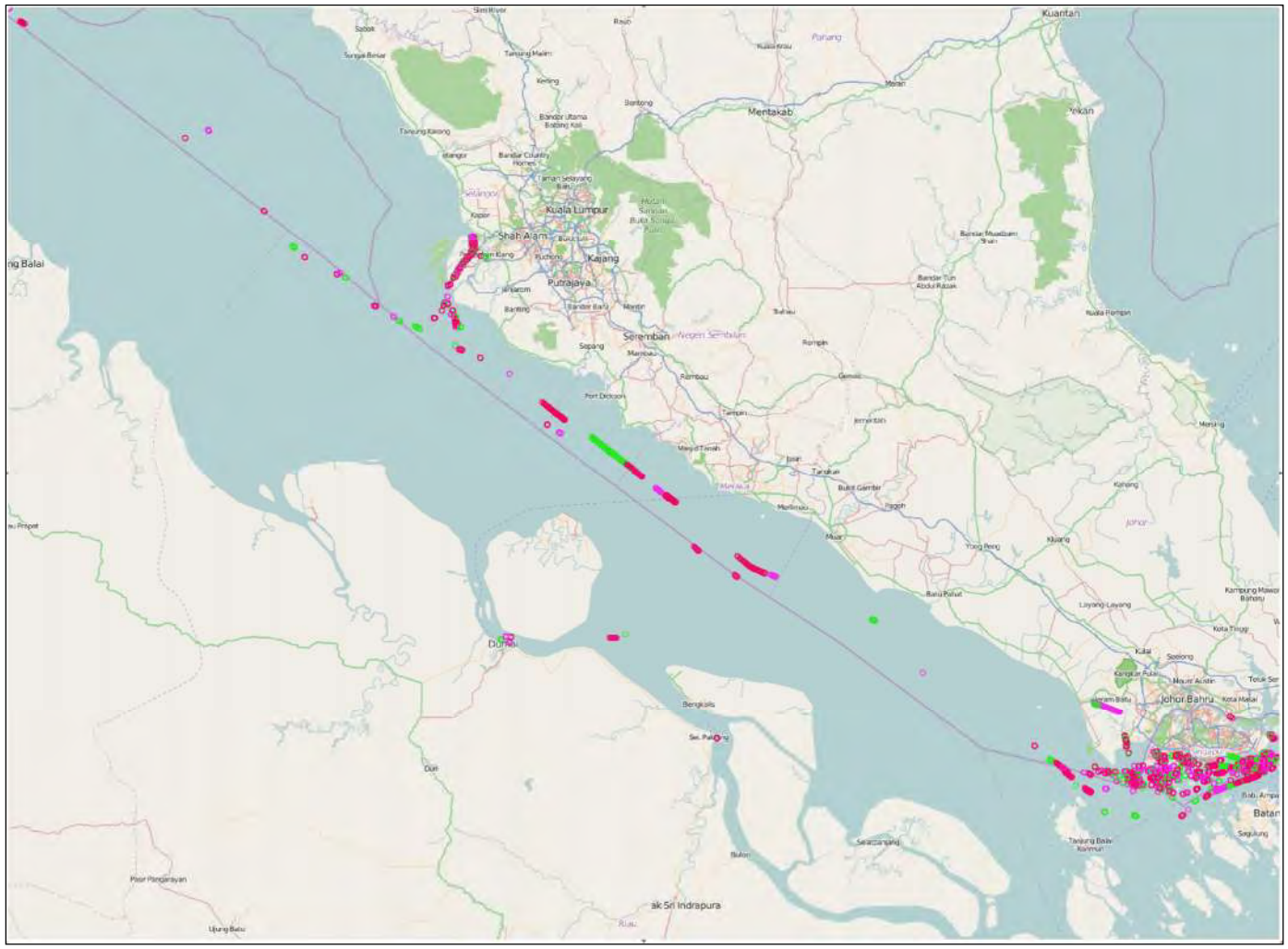

Figure 29. Visualization of encounters February 15, 2011 (Config 2).

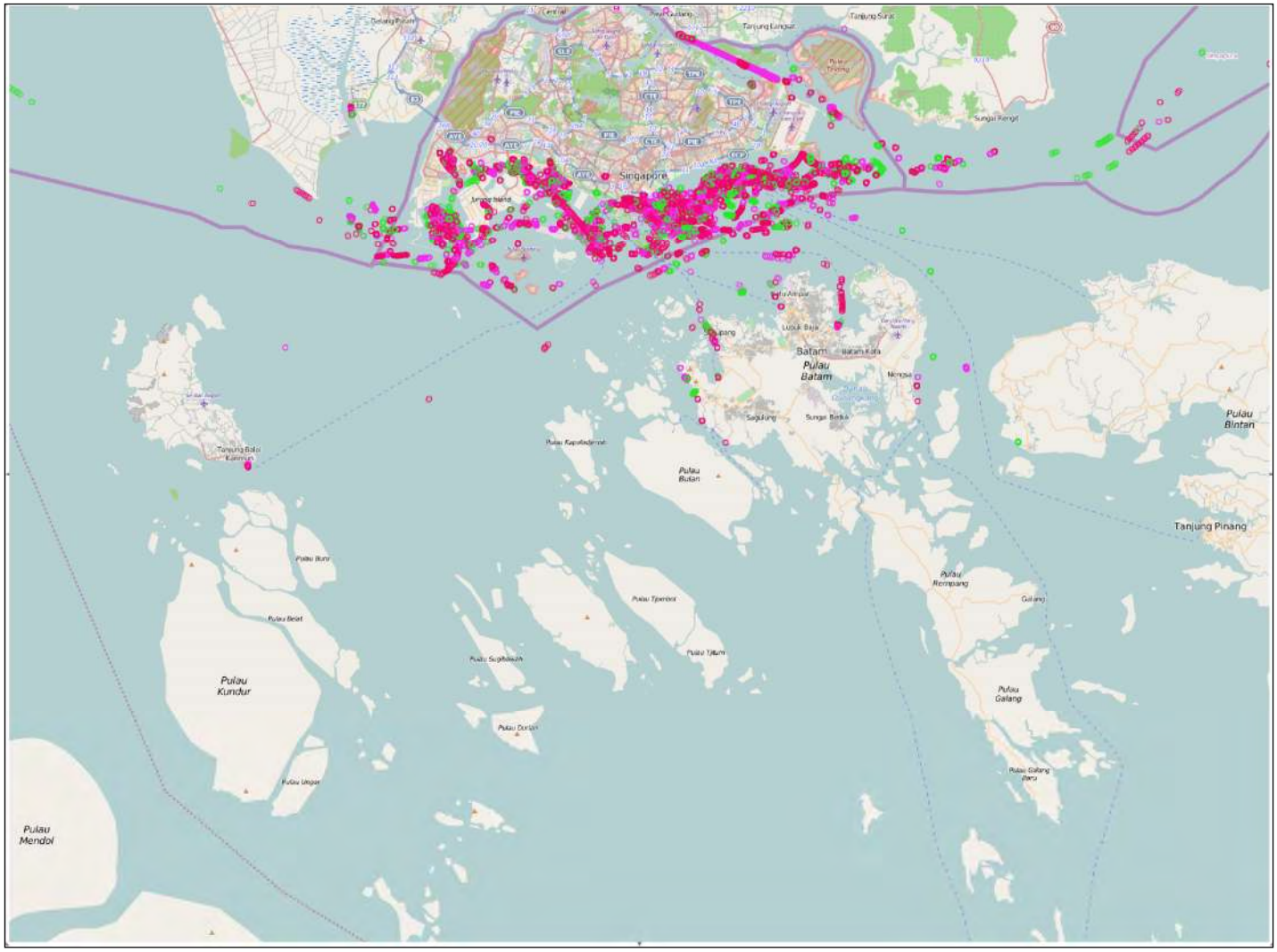

Figure 30. Visualization of encounters February 16, 2011 (Config 1).

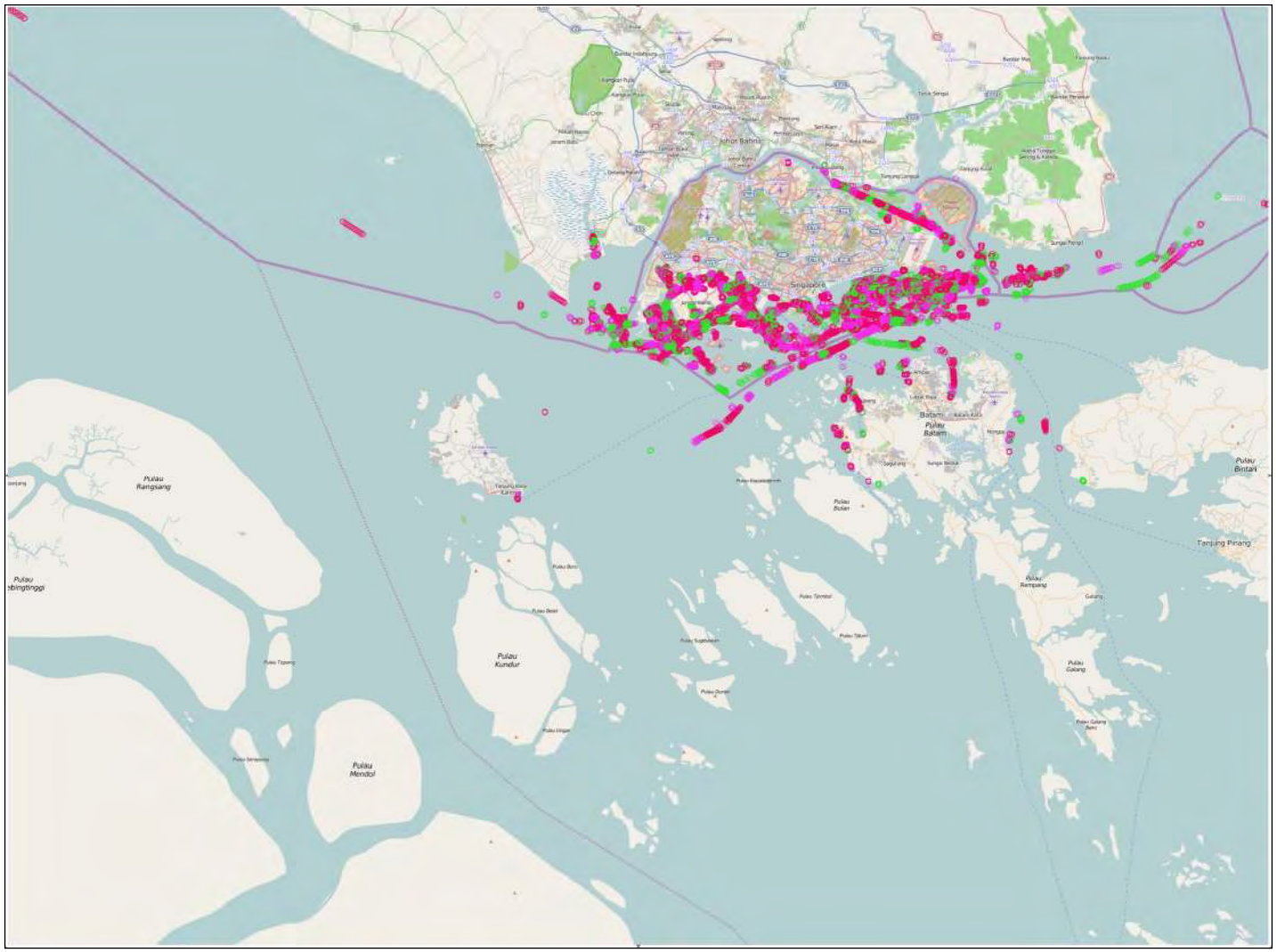

Figure 31. Visualization of encounters February 16, 2011 (Config 2).

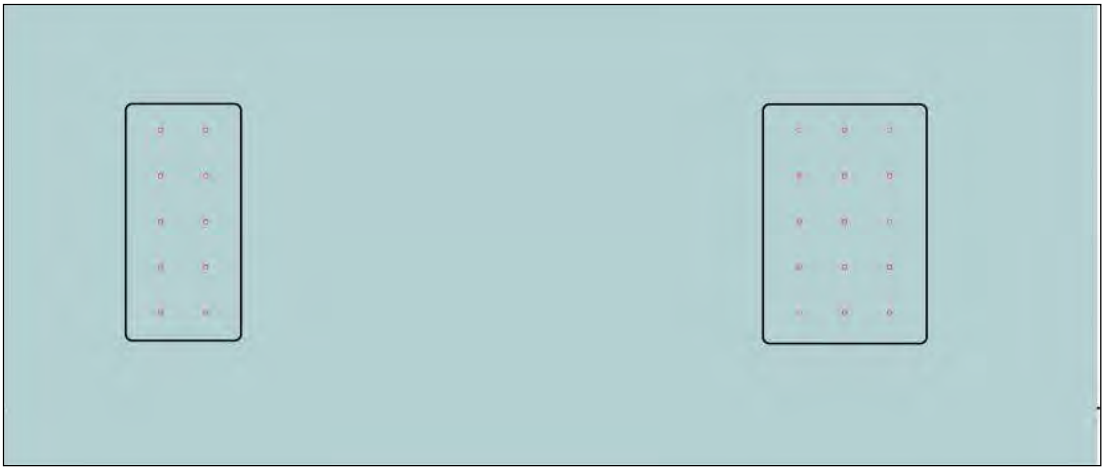

Figure 32. Example of parallel or head front encounter pattern.

The second encounter area indicates a crossing pattern and is shown in Figure 33. Once again each vessel involved in shown with a black box around the trajectory points. The top vessel heading in a horizontal direction is shown in a purple color, while the bottom vessel is heading in a vertical direction and is shown in a red color. Once again the direction of movement is not shown, so it is possible these vessels were moving away from one another; however, it is more likely that the top vessel is traveling either east or west and the bottom vessel is approaching from the south and traveling north.

For the January 11, 2012 data set, an area can be observed that appears to involve an interaction between four different vessels; however, the interaction between these vessels is complicated and requires further investigation. The trajectories of the vessels shown on the left and the vessel shown on the right may be the same vessel. The trajectory formed by the vessel on the left and the trajectory formed by the vessel on the right appear to form a single continuous trajectory. While there appears to be continuity in the trajectory of the upper trajectories, there is no continuity between the trajectory points shown in the lower left and the trajectory points shown on the right. This apparently anomalous behavior is shown in Figure 34.

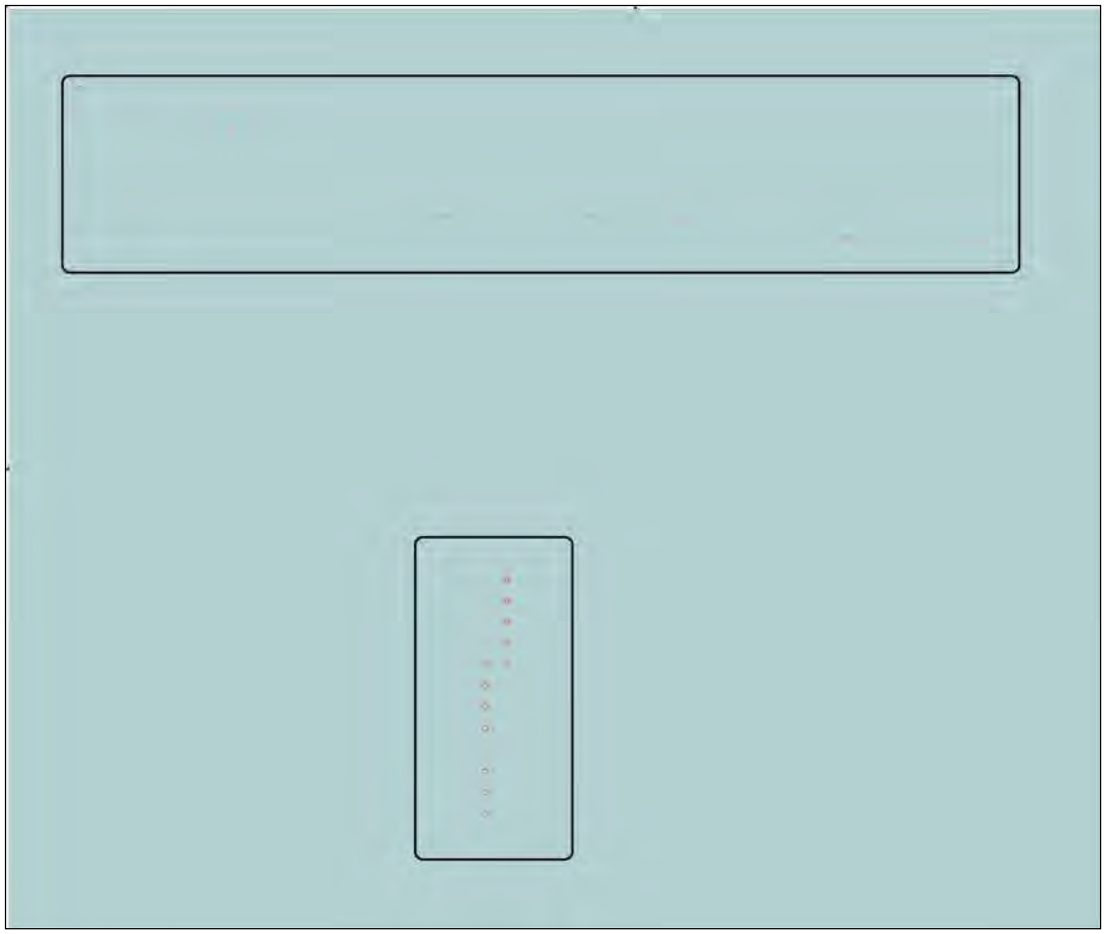

Figure 33. Example of cross-encounter pattern.

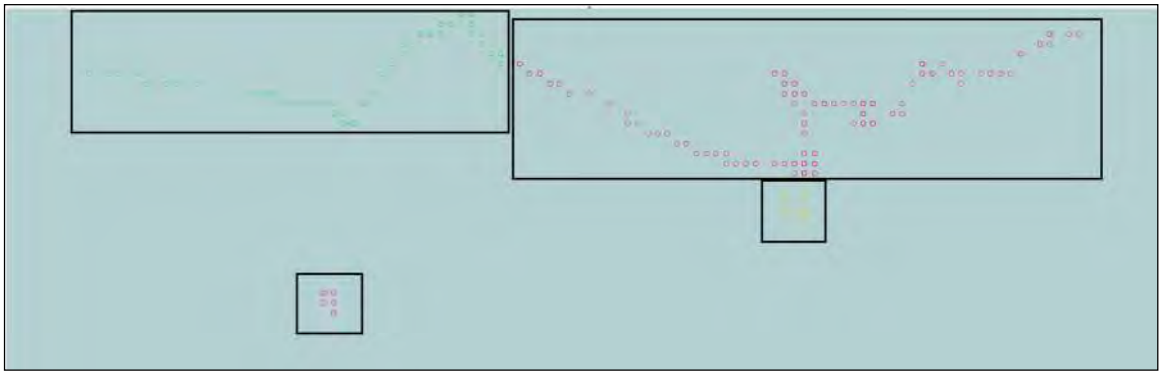

Figure 34. Example of encounter pattern involving multiple vessels.

3. Summary of Results

In this section, patterns on life and potential anomalous activities were investigated using encounter visualizations. In the first subsection it was established that the Config 2 processing values are sufficient for generating visualizations. Visualizations generated using Config 1 were missing some of the encounter detections, while Config 3 and Config 4 did not generate any additional encounter areas to investigate.

In the second subsection, potential anomalous activities were investigated using the encounter visualizations and zooming in on encounter areas of interest. Potential encounter patterns were identified for several example cases. For these example cases, initial observations were recording using the zoomed in encounter visualizations. Additional approaches for anomaly investigation are suggested in the section on future work in the final chapter.

V. CONCLUSIONS

A visual analytics process to detect encounters between vessels from ship positions was developed in this thesis. Research for this thesis supports the development of ABI tradecraft in support of the USN MDA concept. Specifically, this research demonstrated the use of an encounter detection algorithm for patterns of life analysis and to help identify encounter areas to investigate potential anomalous activity. Improvements to the original encounter detection algorithm to enable processing large data sets were also investigated.

Large data sets from a historical AIS archive were first pre-processed in order to decode position records from the original AIS message data. Pre-processing also performed a geographic filter on the data in order to focus the inquiry on a specific geographic area of interest. In this research, the focus was on the TBA of Southeast Asia due to a history of piracy in this region.

Positions records were processed using an encounter detection algorithm developed in MATLAB. The algorithm processed position record data using four different configurations, where each configuration used different values for spatial and temporal sensitivities. Two months of historical data for each configuration were processed using both the original algorithm and a modified algorithm using a sliding spatial window to improve performance when processing large data sets. Both algorithms generated the same number of encounter detections, which demonstrated that both algorithms are equivalent.

Data visualizations were also created using output files generated by the encounter detection algorithm. Data visualizations were explored using the V-Analytics software and the simple process of zooming in order to explore encounter areas of interest.

A. SIGNIFICANT CONTRIBUTIONS

The most significant contribution from this research is the development of an encounter detection algorithm for processing large data sets from a historical AIS

archive. Previous research using the encounter detection algorithm used smaller data sets and in many cases used simulated data. In this research, several key findings were developed regarding the performance of the algorithm when processing large data sets.

The first finding was the growth in memory requirements for the original algorithm when interpolation of data points was required. The original algorithm used a recursive process to recursively call the detection function following the creation of new position records from interpolation. Preliminary work was done in MATLAB using standard procedural calls, and the recursive calls would cause MATLAB to exit processing with an out-of-memory error. Although MATLAB does not support programming with pointers, a pass-by-reference capability can be invoked using the object oriented syntax in MATLAB. This change allowed the original algorithm to complete but did not address the growth in processing time.

The second finding was that the longer the algorithm was running, the slower the encounter detection process became. This was also determined to be caused by the interpolation process. The creation of new positions during the interpolation process was creating an ever increasing number of positions to be checked by the encounter detection function. A pre-interpolation process was introduced to eliminate the processing growth associated with the integrated interpolation process. The addition of the pre-interpolation process to the algorithm created predictable processing times that can be estimated as a function of how many position records are being processed when the number of position records used are the number of total records following interpolation.

The third finding was the modification of the algorithm to add the innovation of a sliding spatial window to complement the sliding temporal window. Pre-interpolation of the positions records allowed for the development of a method using a sliding spatial window. Preliminary results show potential for processing performance improvement with large and very large data sets.

A secondary contribution is the preliminary work done in pattern of life analysis and identification of potentially anomalous behavior using historical AIS data. Most of the data sets did not provide anything potentially of interest when viewing the

visualization; however, in the January 2012 data set there were several days where the visualizations did indicate multiple encounter areas to investigate. Since the algorithm did not generate encounter areas of interest for each data set, it may be useful in building an anomaly detection system. Assuming the algorithm continues to generate a limited number of encounter areas of interest, a human analyst could very quickly investigate these anomalies once cued by the encounter detection algorithm.

B. RECOMMENDATIONS FOR FUTURE WORK

In this thesis research the focus was on development of the encounter detection algorithm in MATLAB and the investigation of possible performance improvements when processing large data sets. AIS data from January 2013 consists of data sets that are larger and generate significantly more position records than either the data processed from February 2011 or January 2012. Future work could investigate processing and analyzing additional data including larger data sets.

In addition to processing more data sets, additional work could be done in exploratory data analysis. Additions to the visual analytics process could be made, including the use of contextual data when investigating encounter areas. Contextual data could be filtered in order to focus on or exclude vessels based on some characteristic defined in the AIS messages. For example, vessels could be screened based on the flag they operate under or the vessel type. The encounter detection algorithm could be expanded to include automatic characterization of encounter patterns to aid in identifying entities, activities and transactions.

Future work could also involve using the encounter detection algorithm with data from another source. Radar position reports could be incorporated along with AIS reports to provide a multi-INT approach to encounter detection.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

The MATLAB code used to implement the encounter detection processing and formatting for post-processing is provided in this appendix.

main.m

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

maxconfig = 4; % 4 different combinations of N (time) and delS (space)

% initialize stats
numRecords = 0;

memoryusage = zeros( 1, maxconfig );
grandtotal = zeros( 1, maxconfig );
numProcessed = zeros( 1, maxconfig );
numEncounters = zeros( 1, maxconfig );
numGroups = zeros( 1, maxconfig );

% pre-interpolation processing

if( ~( exist( 'data', 'dir' ) == 7 ) )

    disp( 'creating data directory...' )
    mkdir data

end

for config = 1 : maxconfig

    starttime( config ) = tic; % record total time
    totalstart = uint64( starttime( config ) );
    disp( [ 'Processing Config #:  ' num2str( config ) ] )

    if( mod( config, 2 ) == 1 ) % clear data for new interpolation

        clear stat

    end

    clear data

    [ N, delS ] = configSetup( config ); % retrieve sensitivity settings

    % intialize log file for storing position records
    if( exist( 'log', 'var' ) == 1 )

        % log file already in workspace - useful for debugging
        disp( 'Log file in workspace is being used.' )

    else

        log = createLog( config, totalstart ); % create new log from AIS log data files;
        endoflog = log.recordlistend;

    end

    if( exist( 'stat', 'var' ) == 1 )
```

```
        % stat file already in workspace - useful for debugging
        disp( 'Statistics found.. skipping interpolation.' )

    else

        stat = interpolateData( log, endoflog, N, config, totalstart ); % interpolate
data before encounter detection
        numRecords = stat.recproc;

    end

    numProcessed( config ) = stat.recproc + stat.totalpos;

    if( exist( 'data', 'var' ) == 1 )

        % stat file already in workspace - useful for debugging
        disp( 'Encounter data found.. skipping encounter detection.  (Debug only)' )

    else

        [ data, statenc ] = detectEncounter( log, N, delS, config, totalstart );

    end

    sdata = mergeEncounters( data, config, totalstart );

    numEncounters( config ) = sdata.totalencounters;
    numGroups( config ) = sdata.groups;

    writeEncounters( sdata, config );

    [ uv, sv ] = memory;
    memoryusage( config ) = round( uv.MemUsedMATLAB / 2^20 ); % in megabytes

    grandtotal( config ) = round( ( toc( totalstart ) / 3600 ), 2 ); % in hours
    disp( [ 'Number of records in record set: ' num2str( numRecords ) ] )
    for gidx = 1 : maxconfig

        disp( '------------------------------------------------------------' )
        disp( [ 'CONFIGURATION #: ' num2str( gidx ) ] )
        disp( [ 'Number of records processed for config #: ' num2str( numProcessed( gidx
) ) ] )
        disp( [ 'Number of encounters: ' num2str( numEncounters( gidx ) ) ] )
        disp( [ 'Number of encounter groups: ' num2str( numGroups( gidx ) ) ] )
        disp( [ 'Grand Total time in hours: ' num2str( grandtotal( gidx ) ) ] )
        disp( [ 'MATLAB memory usage (MB): ' num2str( memoryusage( gidx ) ) ] )
        disp( '------------------------------------------------------------' )

    end

    clear sdata

end
```

configSetup.m

```
function [ N, delS ] = configSetup( config )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% configSetup returns the N (temporal) and delS (spatial) sensitivities

switch( config )

    case 1

        N = 60;
        delS = 185;

    case 2

        N = 60;
        delS = 185 * 2;

    case 3

        N = 30;
        delS = 185;

    case 4

        N = 30;
        delS = 185 *2;
end

end
```

createLog.m

```
function [ log ] = createLog( config, totalstart )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% createLog creates log object from AIS messages (1,2 and 3)

disp( 'Log file is being created in workspace.' )
    log = Log; % create object for AIS log data

    log.recordlist = cell( 1, 100000 );
    log.recordlistend = 0;

    logname   = 'outputm1.csv';

    if( exist( logname, 'file' ) == 2 )

        fid = fopen( logname ); % open file
        disp( 'Reading type 1 messages' )
        log.initialize( fid, config, totalstart ); % read file and create records in log
        fclose( fid );

    else

        disp( 'No type 1 messages found' )

    end
```

```
    logname   = 'outputm2.csv';

    if( exist( logname, 'file' ) == 2 )

        fid = fopen( logname ); % open file
        disp( 'Reading type 2 messages' )
        log.initialize( fid, config, totalstart ); % read file and create records in log
        fclose( fid );

    else

        disp( 'No type 2 messages found' )

    end

    logname   = 'outputm3.csv';

    if( exist( logname, 'file' ) == 2 )

        fid = fopen( logname ); % open file
        disp( 'Reading type 3 messages' )
        log.initialize( fid, config, totalstart ); % read file and create records in log
        fclose( fid );

    else

        disp( 'No type 3 messages found' )

    end

end
```

@Log/Log.m

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

classdef Log < handle
    % Log class data structure

    properties
        recordlist
        recordlistend
        currentrecord
        minlat = -10;
        minlon = 90
        maxlat = 8;
        maxlon = 125;
        recordlimit = 1000;
        divideby = 10;
        preintrecordlist
        preintrecordlistend

    end

    methods

        function lsize = latsize( obj )
            lsize = 1 + ( obj.maxlat - obj.minlat );
        end

        function lsize = lonsize( obj )
            lsize = 1 + ( obj.maxlon - obj.minlon );
        end

        function offset = latoffset( obj )
            offset = -1 * ( obj.minlat - 1 );
        end
```

```
        function offset = lonoffset( obj )
            offset = -1 * ( obj.minlon -1 );
        end

        function index = latidxstart( obj )
            index = obj.latoffset + obj.minlat;
        end

        function index = lonidxstart( obj )
            index = obj.lonoffset + obj.minlon;
        end

        function index = latidxend( obj )
            index = 1 + ( obj.maxlat - obj.minlat );
        end

        function index = lonidxend( obj )
            index = 1 + ( obj.maxlon - obj.minlon );
        end

        function index = latidx( obj, record )
            index = obj.latoffset + floor( record.Lat );
        end

        function index = lonidx( obj, record )
            index = obj.lonoffset + floor( record.Long );
        end

        function index = latdividx( obj, record )
            index = 1 + floor( 10 * ( abs( record.Lat ) - fix( abs( record.Lat ) ) ) );
        end

        function index = londividx( obj, record )
            index = 1 + floor( 10 * ( abs( record.Long ) - fix( abs( record.Long ) ) ) );
        end

       function index = latsubdividx( obj, record )
            index1 = 10 * ( abs( record.Lat ) - fix( abs( record.Lat ) ) );
            index = 1 + floor( 10 * ( abs( index1 ) - fix( abs( index1 ) ) ) );
        end

        function index = lonsubdividx( obj, record )
            index1 = 10 * ( abs( record.Long ) - fix( abs( record.Long ) ) );
            index = 1 + floor( 10 * ( abs( index1 ) - fix( abs( index1 ) ) ) );

        end

        function index = latsubdiv2( obj, record )
            index2 = 10 * ( abs( record.Lat ) - fix( abs( record.Lat ) ) );
            index1 = 10 * ( abs( index2 ) - fix( abs( index2 ) ) );
            index = 1 + floor( 10 * ( abs( index1 ) - fix( abs( index1 ) ) ) );
        end


        function index = lonsubdiv2( obj, record )
            index2 = 10 * ( abs( record.Long ) - fix( abs( record.Long ) ) );
            index1 = 10 * ( abs( index2 ) - fix( abs( index2 ) ) );
            index = 1 + floor( 10 * ( abs( index1 ) - fix( abs( index1 ) ) ) );

        end


    initialize( obj, fid, config, totalstart )

    [ record, maxtime ] = readpreintRecord( obj, splitlat, splitlon, splitlatidx,
splitlonidx, idx )
    end
```

```
end
```

@Log/initialize.m

```
function [ ] = initialize( log, fid, config, totalstart )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% intialize is a method for the Log class

% additional fields in AIS message are commented out to reduce data
% structure size for interpolation and encounter detection

recordcount = 0;

while( ~feof( fid ) )

    % Reads a new record from the input log file
    recordcount = recordcount + 1; % increment record count

    % get UTC field from record
    text = textscan( fid, '%s', 1, 'Delimiter', ',' ); % cell array

    % check for blank lines at end of file
    try

        UTC = text{ 1 }{ 1 }; % dereference cell array and string to char

    catch

        break; % if can't read text assume at end of file

    end
    % extract date and time from UTC

    text = textscan( UTC, '%s', 2 );
    mdy = text{ 1 }{ 1 };
    hms = text{ 1 }{ 2 };

    text = textscan( mdy , '%d', 'Delimiter', '/' );
    month = text{ 1 }( 2 ); % imported using european date format
    day = text{ 1 }( 1 ); % imported using european date format
    year = text{ 1 }( 3 );

    text = textscan( hms, '%d', 'Delimiter', ':' );
    hour = text{ 1 }( 1 );
    minute = text{ 1 }( 2 );
    second = text{ 1 }( 3 );

    % get AIS message type

    text = textscan( fid, '%d', 1, 'Delimiter', ',' );
    AISType = text{ 1 };

    switch ( AISType )

        case{ 1, 2, 3 }

        % create new record (position record)
        record = Record;
        % record.AISType = AISType;

        % read MMSI unique identifier
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        record.MMSI = text{ 1 };
```

```
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        % record.MMSIcode = text{ 1 };
        text = textscan( fid, '%s', 1, 'Delimiter', ',' );
        % record.MMSIregion = text{ 1 };

        % read Navigation info
        text = textscan( fid, '%d', 1, 'Delimiter' , ',' );
        % record.Navnum = text{ 1 };
%         switch( record.Navnum )
%             case( 0 )
%                 record.NavStatus = 'Under way using engine';
%             case( 1 )
%                 record.NavStatus = 'At anchor';
%             case( 2 )
%                 record.NavStatus = 'Not under command';
%             case( 3 )
%                 record.NavStatus = 'Restricted manueverability';
%             case( 4 )
%                 record.NavStatus = 'Moored';
%             case( 5 )
%                 record.NavStatus = 'Aground';
%             case( 6 )
%                 record.NavStatus = 'Engaged in Fishing';
%             case( 7 )
%                 record.NavStatus = 'Under way sailing';
%             case( 8 )
%                 record.NavStatus = 'Under way using engine';
%             otherwise
%                 record.NavStatus = 'Undefined';
%         end

        % read rate of turn
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        % record.ROT = text{ 1 };
%         [a,b] = size( record.ROT );
%
%         if ( a > 1 || b > 1 )
%             disp( ' size problem ' )
%         end
%
%         if( record.ROT ) == 0
%             record.Turn = 'Not turning';
%         elseif( record.ROT > 0  && record.ROT < 127 )
%             record.Turn = [ 'Right at ' num2str( ( abs( record.ROT ) / 4.733 ) ^2 ) '
deg/min' ];
%         elseif( record.ROT < 0 && record.ROT > -127 )
%             record.Turn = [ 'Left at ' num2str( ( abs( record.ROT ) / 4.733 ) ^2 ) '
deg/min' ];
%         elseif( record.ROT == 127 )
%             record.Turn = 'Turning right at more than 708 deg/min';
%         elseif( record.ROT == -127 )
%             record.Turn = 'Turning left at more than 708 deg/min';
%         else
%             record.Turn = 'No turn information available';
%         end

        % read speed over ground
        text = textscan( fid, '%f', 1, 'Delimiter', ',' );
        % record.SOG = text{ 1 }; %knotts

        % position accuracy
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        % record.posacc = text{ 1 };

        % read longitude and lattitude
        text = textscan( fid, '%f', 1, 'Delimiter', ',' );
        record.Long = text{ 1 };%minutes
        text = textscan( fid, '%f', 1, 'Delimiter', ',' );
        record.Lat = text{ 1 };%degrees
```

```
        % read course over ground
        text = textscan( fid, '%f', 1, 'Delimiter', ',' );
        % record.COG = text{ 1 };% degrees - relative to true north

        % read true heading
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        % record.HDG = text{ 1 }; % degrees - true heading

        % read UTC second
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        % record.UTCs = text{ 1 };

        % read version
        text = textscan( fid, '%d', 1, 'Delimiter', ',' );
        % record.ver = text{ 1 };

        % save UTC time stamp
        record.month = month;
        record.day = day;
        record.year = year;

        record.hour = hour;
        record.minute = minute;
        record.second = second;

        otherwise
            % ignore AIS message not 1,2 or 3 - find new line
            textscan( fid, '%*[^\n]', 1, 'Delimiter', ',' );
    end

    log.recordlistend = log.recordlistend + 1;
    log.recordlist{ log.recordlistend } = record;

    if( recordcount == 1 )

        disp( 'Reading records... please wait.' )

    elseif( mod( recordcount, 1000 ) == 0 )

        totaltime = toc( totalstart );
        disp( [ 'Read ' num2str( recordcount ) ' records. ' ] )
        disp( [ 'Time elapsed in seconds for config #: ' num2str( config ) ' = ' num2str(
totaltime ) ] )
    end

end
```

@Record/Record.m

```
classdef Record
    % Record is a class for AIS records derived from AIS logs

    properties

  %      AISType
        MMSI
  %      MMSIcode
  %      MMSIregion
  %      Navnum
  %      NavStatus
  %      ROT
  %      Turn
  %      SOG
  %      posacc
        Long
        Lat
  %      COG
```

```
%       HDG
%       UTCs % second
%       ver
       % time stammp
       month
       day
       year
       hour
       minute
       second

    end

    methods

        time = tsec( obj )

    end

end
```

@Record/tsec.m

```
function [ time ] = tsec( obj )
% calculate time in seconds with offset of +1 since no zero index
time = obj.hour * 3600 + obj.minute * 60 + obj.second + 1;

end
```

@interpolateData.m

```
function stat = interpolateData( log, endoflog, N, config, totalstart )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% interpolateData creates additional position records using recursive calls
% to an interpolation only detect function

% MMSI index and count
logMMSIidx = zeros( 1, 10000 );
logMMSIidxend = 0;
MMSIcount = zeros( 1, 10000 );
MMSIcountidx = 0;

% initialize statistics
stat.totalpos = 0;
stat.recproc = 0;

recordcount = 0; % intialize record counter

copyrecordlist = log.recordlist; % create copy of log for processing
copyrecordlistend = endoflog;
log.recordlistend = endoflog;

recordstoprocess = copyrecordlistend; % use end of log as index

if( recordstoprocess > 0 ) % skip processing if no records

    disp(   '**************************************************' )
    disp( [ 'RECORDS TO PROCESS = ' num2str( recordstoprocess ) ] )

    % start interpolation algorithm
    for recidx = 1 : copyrecordlistend
```

```
        [ newRecord ] = copyrecordlist{ recidx }; % read new record

        functioncall = 0; % number of calls to detect function
        recordcount = recordcount + 1;

        timesaved = zeros( 1, 10000 ); % times of records so not duplicated when saving
interpolations
        timesavedidx = 0;

        sdata = AISData; % create new object for interpolating this MMSI
        sdata.N = N; % set temporal sensitivity

        first_time_flag = true; % allows for short loop when detect not called
        interpolate_log = true; % allows interpolation to be skipped for duplicate MMSIs

        recordssaved = 0; % number of new records saved due to interpolation

        for lidx = 1 : logMMSIidxend % for each MMSI already indexed

            if( interpolate_log == true ) % MMSI index not found

                if( newRecord.MMSI == logMMSIidx( lidx ) ) % no need to interpolate

                    interpolate_log = false; % skip processing
                    break

                end

            else

                break % found MMSI index previously - no need to check

            end

        end

        if( interpolate_log == true ) % skip when MMSI found in index

            MMSIrecords = cell( 1, 10000 ); % records for a single MMSI
            MMSIrecordsend = 0;

            for cridx = 1 : copyrecordlistend % build record list for a single MMSI

                if( newRecord.MMSI == copyrecordlist{ cridx }.MMSI ) % MMSI match

                    MMSIrecordsend = MMSIrecordsend + 1;
                    MMSIrecords{ MMSIrecordsend } = copyrecordlist{ cridx }; % copy
record to list

                end

            end

            for stidx = 2 : MMSIrecordsend % insertion sort algorithm adapted from Cormen
[p.18]

                keyvalue = MMSIrecords{ stidx }.tsec; % calls method to calculate time in
seconds
                keyrecord = MMSIrecords{ stidx };

                % insert key into the sorted sequence
                vidx = stidx - 1;

                while( ( vidx > 0 ) && ( MMSIrecords{ vidx }.tsec > keyvalue ) )

                    MMSIrecords{ vidx + 1 } = MMSIrecords{ vidx };
                    vidx = vidx - 1;

                end
```

```
                MMSIrecords{ vidx + 1 } = keyrecord;

            end % end insertion sort algorithm

            for ridx = 1 : MMSIrecordsend

                    logtime = MMSIrecords{ ridx }.tsec;

                if( interpolate_log == false ) % skip processing

                   break

                end

                if( interpolate_log == true )

                    timesavedidx = timesavedidx + 1;
                    timesaved( timesavedidx ) = logtime;

                    sdata.detect( MMSIrecords{ ridx } ); % call interpolation function
                    functioncall = functioncall + 1;
                    first_time_flag = false; % function has been called

               end

            end

        end

        if( first_time_flag == false )

            if( functioncall ~= sdata.totalfunctions ) % new interpolations to process

                [ row, col ] = size( sdata.recordArray );

                recordssaved = 0;

                for tidx = 1 : col % for each new interpolation

                    if( ~isempty( sdata.recordArray{ tidx } ) ) % not empty

                        for sidx = 1 : timesavedidx % check each saved time

                            if( tidx ~= timesaved( sidx ) ) % save only new records

                                log.recordlistend = log.recordlistend + 1;
                                log.recordlist{ log.recordlistend } = sdata.recordArray{
tidx };

                                recordssaved = recordssaved + 1;

                                break
                            end

                        end

                    end

                end

            end

        else

            if( interpolate_log == false )

            else
```

```
                sdata.totalfunctions = sdata.totalfunctions + 1;

            end

        end

        logMMSIidxend = logMMSIidxend + 1; % increment MMSI index
        logMMSIidx( logMMSIidxend ) = newRecord.MMSI; % add to MMSI index

        MMSIcountidx = MMSIcountidx + 1; % increment count index
        MMSIcount( MMSIcountidx ) = sdata.totalfunctions;
        stat.totalpos = stat.totalpos + recordssaved;
        clear sdata % clean up memory

        if( recidx == 1 )

            disp( 'Interpolation in progress... please wait.' )

        elseif( mod( recidx, 1000 ) == 0 )

            disp( '------------------------------------------------------------------------
' )
            disp( [ 'Record #: ' num2str( recidx ) ' of ' num2str( recordstoprocess ) ] )
            disp( '------------------------------------------------------------------------
' )

            totaltime = toc( totalstart );
            disp( [ 'Time elapsed in seconds for config #: ' num2str( config ) ' = '
num2str( totaltime ) ] )

        end

    end

    stat.recproc = stat.recproc + recordcount;

    disp(   '-----------------------------------------------------' )
    disp( [ 'VESSELS = ' num2str( logMMSIidxend ) ] )
    disp(   '-----------------------------------------------------' )
    disp( [ 'TOTAL NEW POSITIONS = ' num2str( stat.totalpos ) ] )
    disp( [ 'TOTAL RECORDS PROCESSED = ' num2str( stat.recproc ) ] )
    disp( [ 'NEW TOTAL FOR LOG = ' num2str( stat.totalpos + stat.recproc ) ] )

end

disp( 'cleaning up memory...' )
clear copylogrecordlist
clear copyrecordlist
clear copylogrecordlistend
clear copyrecordlistend

end
```

@AISData/AISData.m

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

classdef AISData < handle
    % AISData is top level class for encounter detection using AIS logs
    % inherits handle class to provide call by reference (pointer) behavior

    properties
        recordcount = 0; % number of records read from input file
        recordArray
        numEncounters
        % = zeros( 1e2 ); % number of encounters per group
        numEncounterGroups = 0; % number of groups of encounters detected
```

```
        totalNumEncounters = 0;
        totalfunctions = 0;
        N % number of intervals (seconds)
        delS % distance sensitivity in m (185 m = 0.1 nm)
        endofData
        % = zeros( 1, 100000 ) % last entry in column
        % debug
        % = cell( 1, 100000 ) % debug entry for each input record
        MMSIkey
        % = zeros( 1, 100000 ) % key for MMSI lookup
        MMSIkeylast = 0;
        firsttime
        % = zeros( 1, 100000 ) % keyed prevtime for MMSI
        lasttime
        % = zeros( 1, 100000 ) % keyed nexttime for MMSI
        Latfilter = 6;
        Lonfilter = 93;
        % geofilter = true;
        % skippedrecords = 0;
        totaltime = 0;
        MMSIkeylookup
        encounterlist
        encounters = 0;
        encounterchecks = 0;
        totalencounters
        groups
    end

    methods

        [ newrecord, maxtime ] = readNewRecord( obj )
        initialize( obj, fid )
        detect( obj, newRecord )
        edetect( obj, newRecord )

    end

end
```

@AISData/detect.m

```
function [ ] = detect( data, newRecord )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% detect creates new data points when interpolation is required

% calculate time in seconds with offset of +1 since no zero index
timesec = newRecord.tsec; % time in seconds of current record

data.totalfunctions = data.totalfunctions + 1; % increment for function call

% save new record in data
if( data.MMSIkeylast == 0 ) % first record

    data.MMSIkeylast = 1;
    data.MMSIkey( data.MMSIkeylast ) = newRecord.MMSI;
    data.endofData( timesec ) = 1;
    data.recordArray{ 1 , timesec } = newRecord;
    data.firsttime( data.MMSIkeylast ) = timesec;
    data.lasttime( data.MMSIkeylast ) = timesec;

else % not first record
    for kidx = 1 : data.MMSIkeylast

        if( newRecord.MMSI == data.MMSIkey( kidx ) ) % found key
```

```
            data.recordArray{ kidx, timesec } = newRecord;

            if( ( timesec - 1 ) < data.firsttime( kidx ) ) % check for new first time

                data.firsttime( kidx ) = timesec;

            end

            if( ( timesec - 1 ) > data.lasttime( kidx ) ) % check for new last time

                data.lasttime( kidx ) = timesec;

            end

            [ ~, col ] = size( data.endofData );
            if( timesec > col ) % update index for new record when exceed existing times

                data.endofData( timesec ) = kidx;

            end

            break

        end

        if( kidx == data.MMSIkeylast ) % new key needed because not found above

            data.MMSIkeylast = data.MMSIkeylast + 1;
            data.MMSIkey( data.MMSIkeylast ) = newRecord.MMSI;
            data.recordArray{ data.MMSIkeylast , timesec } = newRecord;
            data.endofData( timesec ) = data.MMSIkeylast;
            data.firsttime( data.MMSIkeylast ) = timesec; % save first time
            data.lasttime( data.MMSIkeylast ) = timesec;
        end
    end
end

% determine if interpolation needed.. searches for records with same MMSI

% initialize flags
found_prev_flag = false;
found_next_flag = false;

prevtime = timesec - 1; % check one entry before

% check MMSI key for previous time
firsttime = timesec; % assume first time is now

for kidx = 1 : data.MMSIkeylast % check for first time in MMSI index

    if( newRecord.MMSI == data.MMSIkey( kidx ) )

        if( data.firsttime > 0 ) % zero is unset value

            firsttime = data.firsttime( kidx );
            MMSIkeyidx = kidx;

        end

        break

    end

end

while( prevtime > firsttime ) && ( found_prev_flag == false ) % previous record not found

    if( found_prev_flag == true ) % stop looking when found
```

```
        break

    end

    prevtime = prevtime - 1; % check below lower window for previous records

    [ r, c ] = size( data.recordArray{ MMSIkeyidx, prevtime } );

    if ~( ( r == 0 ) && ( c == 0 ) ) % not empty

        found_prev_flag = true; % found a previous record match

    end

end

% check MMSI key for next time
nexttime = timesec; % assume next time is now

for kidx = 1 : data.MMSIkeylast % check for last time in MMSI index

    if( newRecord.MMSI == data.MMSIkey( kidx ) )

        lasttime = data.lasttime( kidx );
        MMSIkeyidx = kidx;
        break

    end

end

while( nexttime < lasttime ) && ( found_next_flag == false ) % search outside upper
window

    if( found_next_flag == true )

        break

    end

    nexttime = nexttime + 1;

    [ r, c ] = size( data.recordArray{ MMSIkeyidx, nexttime } );

    if ~( ( r == 0 ) && ( c == 0 ) ) % not empty

            found_next_flag = true; % found a next match

    end

end

if ( found_prev_flag == true )

    timediff = double( timesec - prevtime ); % double is fix for error in division

    if( timediff > data.N ) % create interpolated previous point

        factor = 2;
        midpt = double( prevtime + fix( timediff ./ factor ) );
        lat1 = data.recordArray{ MMSIkeyidx, prevtime }.Lat;
        lat2 = newRecord.Lat;
        lon1 = data.recordArray{ MMSIkeyidx, prevtime }.Long;
        lon2 = newRecord.Long;
        newlat = ( lat1 + lat2 ) ./ 2;
        newlon = ( lon1 + lon2 ) ./ 2;

        % copy record into previous time
```

```
        iPrev = newRecord;
        iPrev.Lat = round( newlat, 4 );
        iPrev.Long = round( newlon, 4 );

        % convert back to HMS
        iPrev.hour = fix( ( midpt / 3600 ) );
        iPrev.minute = fix( ( mod( midpt / 60, 60 ) ) );
        iPrev.second = mod( midpt, 60 ) - 1; % remove +1 index

        if( iPrev.second == -1 )

            iPrev.second = 59; % need to decrement minute and/or hour
            if( iPrev.minute == 0 )

                iPrev.hour = iPrev.hour - 1; % decrement hour
                iPrev.minute = 59;
            else

                iPrev.minute = iPrev.minute - 1;

            end

        end

        % Recursive call for interpolation of previous record
        data.totalfunctions = data.totalfunctions + 1; % increment
        data.detect( iPrev );

    end

end

if( found_next_flag == true )

    timediff = double( nexttime - timesec );

    if( timediff > data.N ) % create interpolated next point

        factor = 2;
        midpt = double( nexttime - fix( timediff ./ factor ) );
        lat1 = data.recordArray{ MMSIkeyidx, nexttime }.Lat;
        lat2 = newRecord.Lat;
        lon1 = data.recordArray{ MMSIkeyidx, nexttime }.Long;
        lon2 = newRecord.Long;
        newlat = ( lat1 + lat2 ) ./ 2;
        newlon = ( lon1 + lon2 ) ./ 2;

        % copy record into next time
        iNext = newRecord;
        iNext.Lat = round( newlat, 4 );
        iNext.Long = round( newlon, 4 );

        % convert back to HMS

        iNext.hour = fix( ( midpt / 3600 ) );
        iNext.minute = fix( ( mod( midpt / 60, 60 ) ) );
        iNext.second = mod( midpt, 60 ) - 1; % remove +1 index

        if( iNext.second == -1 )

            iNext.second = 59; % need to decrement minute and/or hour
            if( iNext.minute == 0 )

                iNext.hour = iNext.hour - 1; % decrement hour
                iNext.minute = 59;
            else

                iNext.minute = iNext.minute - 1;
```

```
            end

        end

        % Recursive call for interpolation of next record
        data.totalfunctions = data.totalfunctions + 1; % increment
        data.detect( iNext );

    end
end
end
```

@detectEncounter.m

```
function [ sdata, stat ] = detectEncounter( log, N, delS, config, totalstart )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% detectEncounter detects encounters between records with different MMSIs

disp( 'Time sorting interpolated data... please wait' )

sdata = AISData;
sdata.N = N;
sdata.delS = delS;
sdata.endofData = zeros( 1, 86400 );

stat.totalenc = 0;
stat.totalgrp = 0;
stat.recproc = 0;
endofTime = 0;

for tidx = 1 : log.recordlistend

    sdata.endofData( log.recordlist{ tidx }.tsec ) = sdata.endofData( log.recordlist{
tidx }.tsec ) + 1; % increment
    sdata.recordArray{ sdata.endofData( log.recordlist{ tidx }.tsec ), log.recordlist{
tidx }.tsec } = log.recordlist{ tidx };
    if( log.recordlist{ tidx }.tsec > endofTime )

        endofTime = log.recordlist{ tidx }.tsec;

    end

end

disp( 'Checking for duplicate records... please wait' )

for tidx = 1 : endofTime

    for ridx = 1 : sdata.endofData( tidx ) % each time

        if( ~isempty( sdata.recordArray{ ridx, tidx } ) )

            if( mod( ridx, 1000 ) == 0 )

                disp( [ 'Checking record # : ' num2str( ridx ) ] )

            end

            MMSI = sdata.recordArray{ ridx, tidx }.MMSI;
            Long = sdata.recordArray{ ridx, tidx }.Long;
            Lat = sdata.recordArray{ ridx, tidx }.Lat;
            time = sdata.recordArray{ ridx, tidx }.tsec;

            for didx = ridx + 1 : sdata.endofData( tidx ) - 1;
```

```
                if( ~isempty( sdata.recordArray{ didx, tidx } ) )

                    if( ( MMSI == sdata.recordArray{ didx, tidx }.MMSI ) && ...
                            ( Long == sdata.recordArray{ didx, tidx }.Long ) && ...
                            ( Lat == sdata.recordArray{ didx, tidx }.Lat  ) && ...
                            ( time == sdata.recordArray{ didx, tidx }.tsec ) ) %
duplicate found

                        sdata.recordArray{ didx, tidx }  = []; % delete record


                    end

                end

            end

        end

    end

end

disp( 'Encounter detections in progress... please wait.' )

for tidx = 1 : endofTime % for each time

    for lidx = 1 : sdata.endofData( tidx ) % for each item in list

        if( ~isempty( sdata.recordArray{ lidx, tidx } ) )

            newRecord = sdata.recordArray{ lidx, tidx }; % read new record

            sdata.edetect( newRecord ); % detect encounters for this record

            stat.totalenc = sdata.totalNumEncounters;
            stat.totalgrp = sdata.numEncounterGroups;
            stat.recproc = stat.recproc + 1;

            if( mod( stat.recproc, 1000 ) == 0 )

                disp( '-------------------------------------------------------------------
----' )
                disp( [ 'Record #: ' num2str( stat.recproc ) ' of ' num2str(
log.recordlistend ) ] )
                disp( '-------------------------------------------------------------------
----' )
                disp(   '-----------------------------------------------------' )
                disp( [ 'ENCOUNTERS = ' num2str( sdata.totalNumEncounters ) ] )
                disp( [ 'GROUPS = ' num2str( sdata.numEncounterGroups ) ] )
                disp(   '-----------------------------------------------------' )
                disp( [ 'TOTAL ENCOUNTERS = ' num2str( stat.totalenc ) ] )
                disp( [ 'TOTAL GROUPS = ' num2str( stat.totalgrp ) ] )
                disp(   '-----------------------------------------------------' )

                totaltime = toc( totalstart );
                disp( [ 'Time elapsed in seconds for config #: ' num2str( config ) ' = '
num2str( totaltime ) ] )

            end

        end

    end

end

disp(   '-----------------------------------------------------' )
disp( [ 'ENCOUNTERS = ' num2str( sdata.totalNumEncounters ) ] )
```

```
disp( [ 'GROUPS = ' num2str( sdata.numEncounterGroups ) ] )
disp(   '-------------------------------------------------' )
disp( [ 'TOTAL ENCOUNTERS = ' num2str( stat.totalenc ) ] )
disp( [ 'TOTAL GROUPS = ' num2str( stat.totalgrp ) ] )
disp(   '-------------------------------------------------' )

end
```

@AISData/edetect.m

```
function [ ] = edetect( data, newRecord )

% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% edetect detects encounters between records with different MMSIs

timesec = newRecord.tsec;
k = timesec;
midx = max( 1, k - data.N ); % lower edge of sliding window
nidx = k; % upper edge of sliding window

% search non-empty time sets

for listidx = midx: nidx

    numpositions = data.endofData( listidx );

    if( numpositions > 0 )

        for posidx = 1 : numpositions

            [ r, c ] = size( data.recordArray{ posidx, listidx } );
            if ~( ( r == 0 ) && ( c == 0 ) ) % not empty

                MMSIi = newRecord.MMSI;
                MMSIj = data.recordArray{ posidx, listidx }.MMSI;

                if ( MMSIi ~= MMSIj ) %  not same vessel

                    data.encounterchecks = data.encounterchecks + 1;

                    lat1 = newRecord.Lat;
                    lat2 = data.recordArray{ posidx, listidx }.Lat;
                    lon1 = newRecord.Long;
                    lon2 = data.recordArray{ posidx, listidx }.Long;

                    meanradius = 6371.009 * 1e3; % mean radius of earth
                    distv = distance( lat1, lon1, lat2, lon2, meanradius );

                     if ( distv <= data.delS ) % detect encounters

                        data.encounters = data.encounters + 1;

                        if( data.numEncounterGroups == 0 ) % first encounter
                            data.numEncounterGroups = 1;
                            encgroup = 1; % set group
                            encposition = 1; % set position within group

                        else % not first encounter

                            match_flag = 0; % initialize flag
                            for encgroupidx = 1 : data.numEncounterGroups % search
existing encounters

                                if( ( data.encounterlist{ encgroupidx, 1 ,1 }.MMSI ==
MMSIi || ...
                                        data.encounterlist{ encgroupidx, 1, 1 }.MMSI ==
```

```
MMSIj ) )
                                    if( ( data.encounterlist{ encgroupidx, 1, 2 }.MMSI ==
MMSIi || ...
                                            data.encounterlist{ encgroupidx, 1, 2 }.MMSI
== MMSIj ) )

                                        % found match
                                        encgroup = encgroupidx;
                                        encposition = data.numEncounters( encgroup ) + 1;
% set position to next in group
                                        match_flag = 1; % found match
                                        break; % stop searching

                                    end

                                end

                            end

                            if( match_flag == 0 ) % no match

                                encgroup = data.numEncounterGroups + 1;
                                data.numEncounterGroups = encgroup;
                                encposition = 1;

                            end
                        end

                        % add to encounter list
                        if( MMSIi < MMSIj )

                            data.encounterlist{ encgroup, encposition, 1 } = newRecord;
                            data.encounterlist{ encgroup, encposition, 2 } =
data.recordArray{ posidx, listidx };

                        else

                            data.encounterlist{ encgroup, encposition, 1 } =
data.recordArray{ posidx, listidx };
                            data.encounterlist{ encgroup, encposition, 2 } = newRecord;

                        end

                        data.encounterlist{ encgroup, encposition, 3 } = distv;

                        % update counts
                        data.numEncounters( encgroup ) = encposition;
                        data.totalNumEncounters = data.totalNumEncounters + 1;

                    end
                end
            end
        end
    end
end
```

mergeEncounters.m

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% mergeEncounters identifies and removes duplicate encounters and groups

function [ sdata ] = mergeenc( sdata, config, totalstart )

    totalencountercount = 0;
    [ row col z ] = size( sdata.encounterlist );
```

```
    for ridx = 1 : row % for each group

        disp( [ 'Searching for duplicate encounters in group #: ' num2str( ridx ) ] )

        deleted = 0;

        for cidx = 1 : ( col - 1 )

            if( ~isempty( sdata.encounterlist{ ridx, cidx, 1 } ) )

                MMSIi = sdata.encounterlist{ ridx, cidx, 1 }.MMSI;
                MMSIj = sdata.encounterlist{ ridx, cidx, 2 }.MMSI;
                Longi = sdata.encounterlist{ ridx, cidx, 1 }.Long;
                Longj = sdata.encounterlist{ ridx, cidx, 2 }.Long;
                Lati = sdata.encounterlist{ ridx, cidx, 1 }.Lat;
                Latj = sdata.encounterlist{ ridx, cidx, 2 }.Lat;
                timei = sdata.encounterlist{ ridx, cidx, 1 }.tsec;
                timej = sdata.encounterlist{ ridx, cidx, 2 }.tsec;

                for didx = ( cidx + 1 ) : col

                    if( ~isempty( sdata.encounterlist{ ridx, didx, 1 } ) )

                        if( ( ( ( MMSIi == sdata.encounterlist{ ridx, didx, 1 }.MMSI ) &&
...
                                ( Longi == sdata.encounterlist{ ridx, didx, 1 }.Long ) &&
...
                                ( Lati == sdata.encounterlist{ ridx, didx, 1 }.Lat  ) &&
...
                                ( timei == sdata.encounterlist{ ridx, didx, 1 }.tsec ) )
&& ...
                                ( ( MMSIj == sdata.encounterlist{ ridx, didx, 2 }.MMSI )
&& ...
                                ( Longj == sdata.encounterlist{ ridx, didx, 2 }.Long ) &&
...
                                ( Latj == sdata.encounterlist{ ridx, didx, 2 }.Lat ) &&
...
                                ( timej == sdata.encounterlist{ ridx, didx, 2 }.tsec ) )
) || ...
                                ( ( ( MMSIi == sdata.encounterlist{ ridx, didx, 2 }.MMSI
) && ...
                                ( Longi == sdata.encounterlist{ ridx, didx, 2 }.Long ) &&
...
                                ( Lati == sdata.encounterlist{ ridx, didx, 2 }.Lat ) &&
...
                                ( timei == sdata.encounterlist{ ridx, didx, 2 }.tsec ) )
&& ...
                                ( ( MMSIj == sdata.encounterlist{ ridx, didx, 1 }.MMSI )
&& ...
                                ( Longj == sdata.encounterlist{ ridx, didx, 1 }.Long ) &&
...
                                ( Latj == sdata.encounterlist{ ridx, didx, 1 }.Lat ) &&
...
                                ( timej == sdata.encounterlist{ ridx, didx, 1 }.tsec ) )
) ) % duplicate found

                            % disp( 'Duplicate found. Deleted.' )
                            sdata.encounterlist{ ridx, didx, 1 } = []; % delete encounter
                            sdata.encounterlist{ ridx, didx, 2 } = [];
                            sdata.encounterlist{ ridx, didx, 3 } = 0;
                            deleted = deleted + 1;

                        end

                    end

                end
```

```
            end

        end

        encountercount = 0;

        disp( 'counting encounters...' )
        for idxcnt = 1 : col

            if( ~isempty( sdata.encounterlist{ ridx, idxcnt, 1 } ) )

                encountercount = encountercount + 1;

            end

        end

        disp( [ 'Encounters found in group #: ' num2str( ridx ) ' of ' num2str( row ) ':
' num2str( encountercount ) ] )
        disp( [ 'Encounters deleted: ' num2str( deleted ) ] )
        totalencountercount = totalencountercount + encountercount;

        disp( [ 'Searching for duplicate entries in group #: ' num2str( ridx ) ] )

        deleted = 0;

        for cidx = 1 : ( col - 1 )

            if( ~isempty( sdata.encounterlist{ ridx, cidx, 1 } ) )

                MMSI = sdata.encounterlist{ ridx, cidx, 1 }.MMSI;
                Long = sdata.encounterlist{ ridx, cidx, 1 }.Long;
                Lat = sdata.encounterlist{ ridx, cidx, 1 }.Lat;
                time = sdata.encounterlist{ ridx, cidx, 1 }.tsec;

                for didx = ( cidx + 1 ) : col

                    if( ~isempty( sdata.encounterlist{ ridx, didx, 1 } ) )

                        if( ( MMSI == sdata.encounterlist{ ridx, didx, 1 }.MMSI ) && ...
                                ( Long == sdata.encounterlist{ ridx, didx, 1 }.Long ) &&
...
                                ( Lat == sdata.encounterlist{ ridx, didx, 1 }.Lat  ) &&
...
                                ( time == sdata.encounterlist{ ridx, didx, 1 }.tsec ) ) %
duplicate found

                            % disp( 'Duplicate found. Deleted.' )
                            sdata.encounterlist{ ridx, didx, 1 } = []; % delete encounter
                            sdata.encounterlist{ ridx, didx, 3 } = 0;
                            deleted = deleted + 1;

                        end

                    end

                end

            end

        end

        disp( [ 'Entries deleted: ' num2str( deleted ) ] )

        deleted = 0;

        for cidx = 1 : ( col - 1 )

            if( ~isempty( sdata.encounterlist{ ridx, cidx, 2 } ) )
```

```
                MMSI = sdata.encounterlist{ ridx, cidx, 2 }.MMSI;
                Long = sdata.encounterlist{ ridx, cidx, 2 }.Long;
                Lat = sdata.encounterlist{ ridx, cidx, 2 }.Lat;
                time = sdata.encounterlist{ ridx, cidx, 2 }.tsec;

                for didx = ( cidx + 1 ) : col

                    if( ~isempty( sdata.encounterlist{ ridx, didx, 2 } ) )

                        if( ( MMSI == sdata.encounterlist{ ridx, didx, 2 }.MMSI ) && ...
                                ( Long == sdata.encounterlist{ ridx, didx, 2 }.Long ) &&
...
                                ( Lat == sdata.encounterlist{ ridx, didx, 2 }.Lat  ) &&
...
                                ( time == sdata.encounterlist{ ridx, didx, 2 }.tsec ) ) %
duplicate found

                            % disp( 'Duplicate found. Deleted.' )
                            sdata.encounterlist{ ridx, didx, 2 } = []; % delete encounter
                            sdata.encounterlist{ ridx, didx, 3 } = 0;
                            deleted = deleted + 1;

                        end

                    end

                end

            end

        end

        disp( [ 'Entries deleted: ' num2str( deleted ) ] )
                totaltime = toc( totalstart );
        disp( [ 'Time elapsed in seconds for config #: ' num2str( config ) ' = ' num2str(
totaltime ) ] )

    end
    disp( [ 'Total Encounters found: ' num2str( totalencountercount ) ] )
    sdata.totalencounters = totalencountercount;
    sdata.groups = row;
end
```

writeEncounters.m

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

function [ ] = writeEncounters( data, config )
% Write encounter detections to file

disp( 'Please wait...writing encounter files to disk.' )

fileroot = getFileDate( config );

cd data

fname3 = [ fileroot '.app' ];

APPL_NAME = [ 'APPL_NAME "' fileroot '" ' ];

bcolor1 = repmat( [ 239 6 101;
           246 42 238;
           45 232 42;
           249 6 123 ], 1000, 1 );
```

```
bcolor2 = repmat( [ 214 218 22;
           23 212 66;
           44 82 247;
           242 25 8 ], 1000, 1 );

bground1 = repmat( [ 105 62 107;
            133 107 110;
            90 18 44;
            4 90 50 ], 1000, 1 );

bground2 = repmat( [ 106 70 19;
            58 61 0;
            36 123 13;
            86 61 95 ], 1000, 1 );

[ d1, d2, d3 ] = size( data.encounterlist ); % find out size of matrix

fidw3 = fopen( fname3, 'wt' );

fprintf( fidw3, '%s\n', APPL_NAME );

TERR_NAME = 'TERR_NAME "TBA" ';

fprintf( fidw3, '%s\n', TERR_NAME );

USER_UNIT = 'USER_UNIT "degree"';

fprintf( fidw3, '%s\n', USER_UNIT );

HAS_GEO_COORD = 'HAS_GEO_COORD +';

fprintf( fidw3, '%s\n', HAS_GEO_COORD );

SHOW_LEGEND = 'SHOW_LEGEND +';

fprintf( fidw3, '%s\n', SHOW_LEGEND );

SHOW_LEGEND_SIZE = 'SHOW_LEGEND_SIZE 30';

fprintf( fidw3, '%s\n', SHOW_LEGEND_SIZE );

SHOW_LEGEND_TERRNAME = 'SHOW_LEGEND_TERRNAME +';

fprintf( fidw3, '%s\n', SHOW_LEGEND_TERRNAME );

SHOW_LEGEND_BGCOLOR = 'SHOW_LEGEND_BGCOLOR +';

fprintf( fidw3, '%s\n', SHOW_LEGEND_BGCOLOR );

SHOW_LEGEND_NOBJECTS = 'SHOW_LEGEND_NOBJECTS +';

fprintf( fidw3, '%s\n', SHOW_LEGEND_NOBJECTS );

SHOW_RECORD_PERSISTENT = 'SHOW_RECORD_PERSISTENT -';

fprintf( fidw3, '%s\n', SHOW_RECORD_PERSISTENT );

SHOW_RECORD_TOOLTIP = 'SHOW_RECORD_TOOLTIP +';

fprintf( fidw3, '%s\n', SHOW_RECORD_TOOLTIP );

SHOW_MANIPULATOR = 'SHOW_MANIPULATOR +';

fprintf( fidw3, '%s\n', SHOW_MANIPULATOR );

SHOW_MANIPULATOR_SIZE = 'SHOW_MANIPULATOR_SIZE +';

fprintf( fidw3, '%s\n', SHOW_MANIPULATOR_SIZE );
```

```
APPL_BGCOLOR = 'APPL_BGCOLOR (192,192,192)';

fprintf( fidw3, '%s\n', APPL_BGCOLOR );

for idxd1 = 1 : d1

    idxd2 = 1;

    enc1 = data.encounterlist{ idxd1, idxd2, 1 };
    enc2 = data.encounterlist{ idxd1, idxd2, 2 };

    if( isempty( enc1 ) )

        break

    end

    MMSI1 = enc1.MMSI;
    MMSI2 = enc2.MMSI;

    fname1 = [ fileroot '_enc' num2str( idxd1 ) '-1_' num2str( MMSI1 ) '.csv' ];
    fname2 = [ fileroot '_enc' num2str( idxd1 ) '-2_' num2str( MMSI2 ) '.csv' ];

    TABLEDATA = [ 'TABLEDATA "' fname1 '" "' fname1 '"' ];

    fprintf( fidw3, '%s\n', TABLEDATA );

    FORMAT = 'FORMAT "CSV"';

    fprintf( fidw3, '%s\n', FORMAT );

    DELIMITER = 'DELIMITER ,""';

    fprintf( fidw3, '%s\n', DELIMITER );

    FIELD_NAMES_IN_ROW = 'FIELD_NAMES_IN_ROW 1';

    fprintf( fidw3, '%s\n', FIELD_NAMES_IN_ROW );

    ID_FIELD = 'ID_FIELD "encID"';

    fprintf( fidw3, '%s\n', ID_FIELD );

    NAME_FIELD = 'NAME_FIELD 1';

    fprintf( fidw3, '%s\n', NAME_FIELD );

    X_FIELD = 'X_FIELD "X"';

    fprintf( fidw3, '%s\n', X_FIELD );

    Y_FIELD = 'Y_FIELD "Y"';

    fprintf( fidw3, '%s\n', Y_FIELD );

    zTimeReference = '<TimeReference>';

    fprintf( fidw3, '%s\n', zTimeReference );

    meaning = 'meaning="OCCURRED_AT"';

    fprintf( fidw3, '%s\n', meaning );

    zTime = '"timeMMDDYYYY"="mm/dd/yyyy hh:tt:ss"';

    fprintf( fidw3, '%s\n', zTime );

    attr_name = '"timeMMDDYYYY"';
```

```
fprintf( fidw3, '%s\n', attr_name );

keep_original_columns = 'keep_original_columns=no';

fprintf( fidw3, '%s\n', keep_original_columns );

z_TimeReference = '</TimeReference>';

fprintf( fidw3, '%s\n', z_TimeReference );

zTYPE = 'TYPE POINT';

fprintf( fidw3, '%s\n', zTYPE );

BUILD_MAP_LAYER = 'BUILD_MAP_LAYER +';

fprintf( fidw3, '%s\n', BUILD_MAP_LAYER );

DRAWING = 'DRAWING +';

fprintf( fidw3, '%s\n', DRAWING );

ALLOW_SPATIAL_FILTER = 'ALLOW_SPATIAL_FILTER +';

fprintf( fidw3, '%s\n', ALLOW_SPATIAL_FILTER );

TRANSPARENCY = 'TRANSPARENCY 0';

fprintf( fidw3, '%s\n', TRANSPARENCY );

BORDERS = 'BORDERS +';

fprintf( fidw3, '%s\n', BORDERS );

BORDERW = 'BORDERW 1';

fprintf( fidw3, '%s\n', BORDERW );

HLIGHTEDW = 'HLIGHTEDW 3';

fprintf( fidw3, '%s\n', HLIGHTEDW );

SELECTEDW = 'SELECTEDW 3';

fprintf( fidw3, '%s\n', SELECTEDW );

BORDERCOLOR = [ 'BORDERCOLOR (' num2str( bcolor1( mod( idxd1, 4000 ) + 1, 1 ) ) ...
    ',' num2str( bcolor1( mod( idxd1, 4000 ) + 1, 2 ) ) ...
    ',' num2str( bcolor1( mod( idxd1, 4000 ) + 1, 3 ) ) ')' ];

fprintf( fidw3, '%s\n', BORDERCOLOR );

BACKGROUND = [ 'BACKGROUND (' num2str( bground1( mod( idxd1, 4000 ) + 1, 1 ) ) ...
    ',' num2str( bground1( mod( idxd1, 4000 ) + 1, 2 ) ) ...
    ',' num2str( bground1( mod( idxd1, 4000 ) + 1, 3 ) ) ')' ];

fprintf( fidw3, '%s\n', BACKGROUND );

HATCH_STYLE = 'HATCH_STYLE 0';

fprintf( fidw3, '%s\n', HATCH_STYLE );

% Repeat for second in pair

TABLEDATA = [ 'TABLEDATA "' fname2 '" "' fname2 '"' ];

fprintf( fidw3, '%s\n', TABLEDATA );

DELIMITER = 'DELIMITER ,""';
```

```
fprintf( fidw3, '%s\n', DELIMITER );
FIELD_NAMES_IN_ROW = 'FIELD_NAMES_IN_ROW 1';
fprintf( fidw3, '%s\n', FIELD_NAMES_IN_ROW );
ID_FIELD = 'ID_FIELD "encID"';
fprintf( fidw3, '%s\n', ID_FIELD );
NAME_FIELD = 'NAME_FIELD 1';
fprintf( fidw3, '%s\n', NAME_FIELD );
X_FIELD = 'X_FIELD "X"';
fprintf( fidw3, '%s\n', X_FIELD );
Y_FIELD = 'Y_FIELD "Y"';
fprintf( fidw3, '%s\n', Y_FIELD );
zTimeReference = '<TimeReference>';
fprintf( fidw3, '%s\n', zTimeReference );
meaning = 'meaning="OCCURRED_AT"';
fprintf( fidw3, '%s\n', meaning );
zTime = '"timeMMDDYYYY"="mm/dd/yyyy hh:tt:ss"';
fprintf( fidw3, '%s\n', zTime );
attr_name = '"timeMMDDYYYY"';
fprintf( fidw3, '%s\n', attr_name );
keep_original_columns = 'keep_original_columns=no';
fprintf( fidw3, '%s\n', keep_original_columns );
z_TimeReference = '</TimeReference>';
fprintf( fidw3, '%s\n', z_TimeReference );
zTYPE = 'TYPE POINT';
fprintf( fidw3, '%s\n', zTYPE );
BUILD_MAP_LAYER = 'BUILD_MAP_LAYER +';
fprintf( fidw3, '%s\n', BUILD_MAP_LAYER );
DRAWING = 'DRAWING +';
fprintf( fidw3, '%s\n', DRAWING );
ALLOW_SPATIAL_FILTER = 'ALLOW_SPATIAL_FILTER +';
fprintf( fidw3, '%s\n', ALLOW_SPATIAL_FILTER );
TRANSPARENCY = 'TRANSPARENCY 0';
fprintf( fidw3, '%s\n', TRANSPARENCY );
BORDERS = 'BORDERS +';
```

```
fprintf( fidw3, '%s\n', BORDERS );

BORDERW = 'BORDERW 1';

fprintf( fidw3, '%s\n', BORDERW );

HLIGHTEDW = 'HLIGHTEDW 3';

fprintf( fidw3, '%s\n', HLIGHTEDW );

SELECTEDW = 'SELECTEDW 3';

fprintf( fidw3, '%s\n', SELECTEDW );

fprintf( fidw3, '%s\n', BORDERCOLOR );

fprintf( fidw3, '%s\n', BACKGROUND );

HATCH_STYLE = 'HATCH_STYLE 0';

fprintf( fidw3, '%s\n', HATCH_STYLE );

fidw1 = fopen( fname1, 'wt' );
fidw2 = fopen( fname2, 'wt' );

enc1 = data.encounterlist{ idxd1, idxd2, 1 };
enc2 = data.encounterlist{ idxd1, idxd2, 2 };

MMSI1 = enc1.MMSI;
MMSI2 = enc2.MMSI;

enc1ID = [ num2str( MMSI1 ) '_' num2str( idxd1 ) ];
enc2ID = [ num2str( MMSI2 ) '_' num2str( idxd1 ) ];

outarray = 'encID,encN,pldx,X,Y,timeMMDDYYYY';
fprintf( fidw1, '%s\n', outarray );

outarray = 'encID,encN,pldx,X,Y,timeMMDDYYYY';
fprintf( fidw2, '%s\n', outarray );

while( idxd2 <= d2 )

    enc1 = data.encounterlist{ idxd1, idxd2, 1 };

    if( ~isempty( enc1 ) )

        if( enc1.month < 10 )
            monthstr1 = [ '0' num2str( enc1.month ) ];
        else
            monthstr1 = num2str( enc1.month );
        end

        if( enc1.day < 10 )
            daystr1 = [ '0' num2str( enc1.day ) ];
        else
            daystr1 = num2str( enc1.day );
        end

        if( enc1.hour < 10 )
            hourstr1 = [ '0' num2str( enc1.hour ) ];
        else
            hourstr1 = num2str( enc1.hour );
        end

        if( enc1.minute < 10 )
            minutestr1 = [ '0' num2str( enc1.minute ) ];
        else
            minutestr1 = num2str( enc1.minute );
        end
```

```
            if( enc1.second < 10 )
                secondstr1 = [ '0' num2str( enc1.second ) ];
            else
                secondstr1 = num2str( enc1.second );
            end

            outarray = [ num2str( enc1ID ) ',' num2str( 1 ) ',' num2str( idxd2 ) ',' ...
            num2str( enc1.Long ) ',' num2str( enc1.Lat ) ',' ...
            monthstr1 '/' daystr1 '/' num2str( enc1.year ) ...
            ' ' hourstr1 ':' minutestr1 ':' secondstr1 ];

            fprintf( fidw1, '%s\n', outarray );

        end

        enc2 = data.encounterlist{ idxd1, idxd2, 2 };

        if( ~isempty( enc2 ) )

            if( enc2.month < 10 )
                monthstr2 = [ '0' num2str( enc2.month ) ];
            else
                monthstr2 = num2str( enc2.month );
            end

            if( enc2.month < 10 )
                daystr2 = [ '0' num2str( enc2.day ) ];
            else
                daystr2 = num2str( enc2.day );
            end


            if( enc2.hour < 10 )
                hourstr2 = [ '0' num2str( enc2.hour ) ];
            else
                hourstr2 = num2str( enc2.hour );
            end


            if( enc2.minute < 10 )
                minutestr2 = [ '0' num2str( enc2.minute ) ];
            else
                minutestr2 = num2str( enc2.minute );
            end

            if( enc2.second < 10 )
                secondstr2 = [ '0' num2str( enc2.second ) ];
            else
                secondstr2 = num2str( enc2.second );
            end

            outarray = [ num2str( enc2ID ) ',' num2str( 2 ) ',' num2str( idxd2 ) ',' ...
            num2str( enc2.Long ) ',' num2str( enc2.Lat ) ',' ...
            monthstr2 '/' daystr2 '/' num2str( enc2.year ) ...
            ' ' hourstr2 ':' minutestr2 ':' secondstr2 ];

            fprintf( fidw2, '%s\n', outarray );

        end

        idxd2 = idxd2 + 1;

    end

    fclose( fidw1 );
    fclose( fidw2 );

end
```

```
fclose( fidw3 );

cd ..

end
```

getFileDate.m

```
function [ fileroot ] = getFileDate( config )

% open log file to obtain date of AIS data
fidinit = fopen( 'outputm1.csv' ); % open file

% read current record (line) to obtain UTC date for log file

    % get UTC field from record
    text = textscan( fidinit, '%s', 1, 'Delimiter', ',' ); % cell array
    UTC = text{ 1 }{ 1 }; % dereference cell array and string to char

    % extract date from UTC

    text = textscan( UTC, '%s', 1 );
    mdy = text{ 1 }{ 1 };
    text = textscan( mdy , '%d', 'Delimiter', '/' );
    month = text{ 1 }( 2 ); % imported using european date format
    day = text{ 1 }( 1 ); % imported using european date format
    year = text{ 1 }( 3 );

    % build YMD string

    if( ( month < 10 ) && ( day < 10 ) )
        filedate = strcat( num2str( year ), num2str( 0 ), num2str( month ), num2str( 0 ),
num2str( day ) );
    elseif( ( month < 10 ) && ( day >= 10 ) )
        filedate = strcat( num2str( year ), num2str( 0 ), num2str( month ), num2str( day
) );
    elseif( ( month >= 10 ) && ( day < 10 ) )
        filedate = strcat( num2str( year ), num2str( month ), num2str( 0 ), num2str( day
) );
    else
        filedate = strcat( num2str( year ), num2str( month ), num2str( day ) );
    end

fileroot = [ filedate '-' num2str( config ) ];

fclose( fidinit ); % close file

end
```

detectEncounter.m (spatial window version)

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015
%
% This version includes the use of a spatial window
% Requires spatial window version of mergeEncounters.m

% detectEncounter detects encounters between records with different MMSIs

function [ edata, stat ] = detectEncounter( log, N, delS, config, totalstart )
% detect encounters

switch( config )

    case( 1 )
```

```
        senslowr = 1; % config 1 & 3
        senshighr = 0; % config 1 & 3

    case( 2 )

        senslowr = 2; % config 2 & 4
        senshighr = 1; % config 2 & 4

    case( 3 )

        senslowr = 1; % config 1 & 3
        senshighr = 0; % config 1 & 3

    case( 4 )

        senslowr = 2; % config 2 & 4
        senshighr = 1; % config 2 & 4

end

senshigh = 9;
senslow = 2;

stat.totalenc = 0;
stat.totalgrp = 0;
stat.recproc = 0;

encounterset = 0;

recordsublistend = zeros( log.latsize, log.lonsize );
recordsublist = cell( log.latsize, log.lonsize, 1000 );

disp( '------------------------------------------------------------------------' )
disp( [ 'Time sorting and dividing based on latitude and longitude : ' num2str(
log.recordlistend ) ' records ' ] )
disp( '------------------------------------------------------------------------' )

for tidx = 1 : log.recordlistend;

    if( mod( tidx, 100000 ) == 0 )

        disp( '------------------------------------------------------------------------' )
        disp( [ 'Record #: ' num2str( tidx ) ' of ' num2str( log.recordlistend ) ] )
        disp( '------------------------------------------------------------------------' )

    end

    temp = log.recordlist{ tidx }; % record to be sorted
    latidx = log.latidx( temp );
    lonidx = log.lonidx( temp );
    latdividx = log.latdividx( temp );
    londividx = log.londividx( temp );
    latsubdividx = log.latsubdividx( temp );
    lonsubdividx = log.lonsubdividx( temp );
    latsubdiv2 = log.latsubdiv2( temp );
    lonsubdiv2 = log.lonsubdiv2( temp );
    underlat = false;
    underlon = false;
    overlat = false;
    overlon = false;

    recordsublistend( latidx, lonidx ) = recordsublistend( latidx, lonidx ) + 1;
    recordsublist{ latidx, lonidx, recordsublistend( latidx, lonidx ) } = temp;

    if( ( latsubdiv2 > senshigh - senshighr ) && ( latsubdividx > senshigh ) && (
latdividx > senshigh ) && ( latidx < log.latidxend ) ) % under lat

        recordsublistend( latidx + 1, lonidx ) = recordsublistend( latidx + 1, lonidx ) +
1;
```

```
        recordsublist{ latidx + 1, lonidx, recordsublistend( latidx + 1, lonidx ) } =
temp;
        underlat = true;

    end

    if( ( lonsubdiv2 > senshigh - senshighr ) && ( lonsubdividx > senshigh ) && (
londividx > senshigh ) && ( lonidx < log.lonidxend ) ) % under lon
recordsublistend( latidx, lonidx + 1 ) = recordsublistend( latidx, lonidx + 1 ) + 1;

        recordsublistend( latidx, lonidx + 1 ) = recordsublistend( latidx, lonidx + 1 ) +
1;
        recordsublist{ latidx, lonidx + 1, recordsublistend( latidx, lonidx + 1 ) } =
temp;
        underlon = true;

    end

    if( ( latsubdiv2 < senslow + senslowr ) && ( latsubdividx < senslow ) && ( latdividx
< senslow ) && ( latidx > log.latidxstart ) ) % over lat

        recordsublistend( latidx - 1, lonidx ) = recordsublistend( latidx - 1, lonidx ) +
1;
        recordsublist{ latidx - 1, lonidx, recordsublistend( latidx - 1, lonidx ) } =
temp;
        overlat = true;

    end

    if( ( lonsubdiv2 < senslow + senslowr ) && ( lonsubdividx < senslow ) && ( londividx
< senslow ) && ( lonidx > log.lonidxstart ) ) % over lon

        recordsublistend( latidx, lonidx - 1 ) = recordsublistend( latidx, lonidx - 1 ) +
1;
        recordsublist{ latidx, lonidx - 1, recordsublistend( latidx, lonidx - 1 ) } =
temp;
        overlon = true;

    end

    if( ( overlat == true ) && ( overlon == true ) )

        recordsublistend( latidx - 1, lonidx - 1 ) = recordsublistend( latidx - 1, lonidx
- 1 ) + 1;
        recordsublist{ latidx - 1, lonidx - 1, recordsublistend( latidx - 1, lonidx - 1 )
} = temp;

    elseif( ( underlat == true ) && ( underlon == true ) )

        recordsublistend( latidx + 1, lonidx + 1 ) = recordsublistend( latidx + 1, lonidx
+ 1 ) + 1;
        recordsublist{ latidx + 1, lonidx + 1, recordsublistend( latidx + 1, lonidx + 1 )
} = temp;

    elseif( ( overlat == true ) && ( underlon == true ) )

        recordsublistend( latidx - 1, lonidx + 1 ) = recordsublistend( latidx - 1, lonidx
+ 1 ) + 1;
        recordsublist{ latidx - 1, lonidx + 1, recordsublistend( latidx - 1, lonidx + 1 )
} = temp;

     elseif( ( underlat == true ) && ( overlon == true ) )

        recordsublistend( latidx + 1, lonidx - 1 ) = recordsublistend( latidx + 1, lonidx
- 1 ) + 1;
        recordsublist{ latidx + 1, lonidx - 1, recordsublistend( latidx + 1, lonidx - 1 )
} = temp;

    end
```

```
end

for latidx = log.latidxstart : log.latidxend

    for lonidx = log.lonidxstart : log.lonidxend

        disp( '------------------------------------------------------------------------' )
        disp( [ 'Searching: Lat: ' num2str( latidx ) ', Long: ' num2str( lonidx ) ',' ] )
        disp( '------------------------------------------------------------------------' )

        for latdividx = 1 : log.divideby

            for londividx = 1 : log.divideby

                sdata = AISData;
                sdata.N = N;
                sdata.delS = delS;
                sdata.endofData = zeros( 1, 86400 );

                recordcount = 0;
                endofTime = 0;

                for tidx = 1 : recordsublistend( latidx, lonidx );

                    temp = recordsublist{ latidx, lonidx, tidx }; % record to be sorted

                    % Normal check lat / lon
                    if( ( latdividx == log.latdividx( temp ) ) && ( londividx ==
log.londividx( temp ) ) )

                        sdata.endofData( temp.tsec ) = sdata.endofData( temp.tsec ) + 1;
% increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), temp.tsec } =
temp;

                        recordcount = recordcount + 1;

                        if( temp.tsec > endofTime )

                            endofTime = temp.tsec;

                        end

                    % Four checks with lat normal
                    elseif( ( ( latdividx ) == log.latdividx( temp ) ) && ...
                        ( ( londividx + 1 ) == log.londividx( temp ) ) && ...
                        ( log.lonsubdividx( temp ) < senslow ) && ...
                        ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                        recordcount = recordcount + 1;

                        if( temp.tsec > endofTime )

                            endofTime = temp.tsec;

                        end

                    elseif( ( ( latdividx ) == log.latdividx( temp ) ) && ...
                        ( ( londividx - 1 ) == log.londividx( temp ) ) && ...
                        ( log.lonsubdividx( temp ) > senshigh ) && ...
                        ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                        sdata.endofData( temp.tsec ) = ...
```

```
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( ( latdividx ) == log.latdividx( temp ) ) && ...
                    ( londividx == log.divideby ) && ( log.londividx( temp ) == 1 )
&& ...
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                    sdata.endofData( temp.tsec ) = ...
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( ( latdividx ) == log.latdividx( temp ) ) && ...
                    ( londividx == 1 ) && ( log.londividx( temp ) == log.divideby )
&& ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                    sdata.endofData( temp.tsec ) = ...
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                % five checks with Lat + 1 = record
                elseif( ( ( latdividx + 1 ) == log.latdividx( temp ) ) && ...
                    ( ( londividx ) == log.londividx( temp ) ) && ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end
```

```
                elseif( ( ( latdividx + 1 ) == log.latdividx( temp ) ) && ...
                    ( ( londividx + 1 ) == log.londividx( temp ) ) && ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( ( latdividx + 1 ) == log.latdividx( temp ) ) && ...
                    ( ( londividx - 1 ) == log.londividx( temp ) ) && ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( ( latdividx + 1 ) == log.latdividx( temp ) ) && ...
                    ( londividx == log.divideby ) && ( log.londividx( temp ) == 1 )
&& ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( ( latdividx + 1 ) == log.latdividx( temp ) ) && ...
                    ( londividx == 1 ) && ( log.londividx( temp ) == log.divideby )
&& ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )
```

```
        sdata.endofData( temp.tsec ) = ...
            sdata.endofData( temp.tsec ) + 1; % increment
        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
            temp.tsec } = temp;

    recordcount = recordcount + 1;

    if( temp.tsec > endofTime )

        endofTime = temp.tsec;

    end

% five check with lat - 1 = record
elseif( ( ( latdividx - 1 ) == log.latdividx( temp ) ) && ...
    ( ( londividx ) == log.londividx( temp ) ) && ...
    ( log.latsubdividx( temp ) > senshigh ) && ...
    ( log.latsubdiv2( temp ) > senshigh - senshighr ) )

    sdata.endofData( temp.tsec ) = ...
        sdata.endofData( temp.tsec ) + 1; % increment
    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
        temp.tsec } = temp;

    recordcount = recordcount + 1;

    if( temp.tsec > endofTime )

        endofTime = temp.tsec;

    end

elseif( ( ( latdividx - 1 ) == log.latdividx( temp ) ) && ...
    ( ( londividx + 1 ) == log.londividx( temp ) ) && ...
    ( log.latsubdividx( temp ) > senshigh ) && ...
    ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
    ( log.lonsubdividx( temp ) < senslow ) && ...
    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

    sdata.endofData( temp.tsec ) = ...
        sdata.endofData( temp.tsec ) + 1; % increment
    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
        temp.tsec } = temp;

    recordcount = recordcount + 1;

    if( temp.tsec > endofTime )

        endofTime = temp.tsec;

    end

elseif( ( ( latdividx - 1 ) == log.latdividx( temp ) ) && ...
    ( ( londividx - 1 ) == log.londividx( temp ) ) && ...
    ( log.latsubdividx( temp ) > senshigh ) && ...
    ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
    ( log.lonsubdividx( temp ) > senshigh ) && ...
    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

    sdata.endofData( temp.tsec ) = ...
        sdata.endofData( temp.tsec ) + 1; % increment
    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
        temp.tsec } = temp;

    recordcount = recordcount + 1;

    if( temp.tsec > endofTime )

        endofTime = temp.tsec;
```

```
                    end

                elseif( ( ( latdividx - 1 ) == log.latdividx( temp ) ) && ...
                    ( londividx == log.divideby ) && ( log.londividx( temp ) == 1 )
&& ...
                    ( log.latsubdividx( temp ) > senshigh ) && ...
                    ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                    sdata.endofData( temp.tsec ) = ...
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( ( latdividx - 1 ) == log.latdividx( temp ) ) && ...
                    ( londividx == 1 ) && ( log.londividx( temp ) == log.divideby )
&& ...
                    ( log.latsubdividx( temp ) > senshigh ) && ...
                    ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                    sdata.endofData( temp.tsec ) = ...
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end
                % five check with latdiv 10 = record latdiv = 1
                elseif( ( latdividx == log.divideby ) && ( log.latdividx( temp ) == 1
) && ...
                    ( ( londividx ) == log.londividx( temp ) ) && ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( latdividx == log.divideby ) && ( log.latdividx( temp ) == 1
) && ...
                    ( ( londividx + 1 ) == log.londividx( temp ) ) && ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
```

```
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( latdividx == log.divideby ) && ( log.latdividx( temp ) == 1
) && ...
                    ( ( londividx - 1 ) == log.londividx( temp ) ) && ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( latdividx == log.divideby ) && ( log.latdividx( temp ) == 1
) && ...
                    ( londividx == log.divideby ) && ( log.londividx( temp ) == 1 )
&& ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                        sdata.endofData( temp.tsec ) = ...
                            sdata.endofData( temp.tsec ) + 1; % increment
                        sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                            temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( latdividx == log.divideby ) && ( log.latdividx( temp ) == 1
) && ...
                    ( londividx == 1 ) && ( log.londividx( temp ) == log.divideby )
&& ...
                    ( log.latsubdividx( temp ) < senslow ) && ...
                    ( log.latsubdiv2( temp ) < senslow + senslowr ) && ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                        sdata.endofData( temp.tsec ) = ...
```

```
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                recordcount = recordcount + 1;

                if( temp.tsec > endofTime )

                    endofTime = temp.tsec;

                end

            % five check with latdiv = 1 and record latdiv = 10
            elseif( ( latdividx == 1 ) && ( log.latdividx( temp ) == log.divideby
) && ...
                ( ( londividx ) == log.londividx( temp ) ) && ...
                ( log.latsubdividx( temp ) > senshigh ) && ...
                ( log.latsubdiv2( temp ) > senshigh - senshighr ) )

                sdata.endofData( temp.tsec ) = ...
                    sdata.endofData( temp.tsec ) + 1; % increment
                sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                    temp.tsec } = temp;

                recordcount = recordcount + 1;

                if( temp.tsec > endofTime )

                    endofTime = temp.tsec;

                end

            elseif( ( latdividx == 1 ) && ( log.latdividx( temp ) == log.divideby
) && ...
                ( ( londividx + 1 ) == log.londividx( temp ) ) && ...
                ( log.latsubdividx( temp ) > senshigh ) && ...
                ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
                ( log.lonsubdividx( temp ) < senslow ) && ...
                ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                sdata.endofData( temp.tsec ) = ...
                    sdata.endofData( temp.tsec ) + 1; % increment
                sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                    temp.tsec } = temp;

                recordcount = recordcount + 1;

                if( temp.tsec > endofTime )

                    endofTime = temp.tsec;

                end

            elseif( ( latdividx == 1 ) && ( log.latdividx( temp ) == log.divideby
) && ...
                ( ( londividx - 1 ) == log.londividx( temp ) ) && ...
                ( log.latsubdividx( temp ) > senshigh ) && ...
                ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
                ( log.lonsubdividx( temp ) > senshigh ) && ...
                ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                sdata.endofData( temp.tsec ) = ...
                    sdata.endofData( temp.tsec ) + 1; % increment
                sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                    temp.tsec } = temp;

                recordcount = recordcount + 1;

                if( temp.tsec > endofTime )
```

```
                        endofTime = temp.tsec;

                    end

                elseif( ( latdividx == 1 ) && ( log.latdividx( temp ) == log.divideby
) && ...
                    ( londividx == log.divideby ) && ( log.londividx( temp ) == 1 )
&& ...
                    ( log.latsubdividx( temp ) > senshigh ) && ...
                    ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
                    ( log.lonsubdividx( temp ) < senslow ) && ...
                    ( log.lonsubdiv2( temp ) < senslow + senslowr ) )

                    sdata.endofData( temp.tsec ) = ...
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                elseif( ( latdividx == 1 ) && ( log.latdividx( temp ) == log.divideby
) && ...
                    ( londividx == 1 ) && ( log.londividx( temp ) == log.divideby )
&& ...
                    ( log.latsubdividx( temp ) > senshigh ) && ...
                    ( log.latsubdiv2( temp ) > senshigh - senshighr ) && ...
                    ( log.lonsubdividx( temp ) > senshigh ) && ...
                    ( log.lonsubdiv2( temp ) > senshigh - senshighr ) )

                    sdata.endofData( temp.tsec ) = ...
                        sdata.endofData( temp.tsec ) + 1; % increment
                    sdata.recordArray{ sdata.endofData( temp.tsec ), ...
                        temp.tsec } = temp;

                    recordcount = recordcount + 1;

                    if( temp.tsec > endofTime )

                        endofTime = temp.tsec;

                    end

                end

            end

            if( recordcount > 0 )

                disp(   '-------------------------------------------------' )
                disp( [ 'LATITUDE = ' num2str( latidx - log.latoffset ) '.' num2str(
latdividx - 1 ) ] )
                disp( [ 'LONGITUDE = ' num2str( lonidx - log.lonoffset ) '.' num2str(
londividx - 1 ) ] )
                disp( [ 'RECORDS TO CHECK: ' num2str( recordcount ) ] )
                disp(   '-------------------------------------------------' )

                stat.recproc = 0;

                disp( 'Checking for duplicate records... please wait' )

                for tidx = 1 : endofTime
```

```
                    for ridx = 1 : ( sdata.endofData( tidx ) - 1 ) % each time

                        if( ~isempty( sdata.recordArray{ ridx, tidx } ) )

                            if( mod( ridx, 1000 ) == 0 )

                                disp( [ 'Checking record # : ' num2str( ridx ) ] )

                            end

                            MMSI = sdata.recordArray{ ridx, tidx }.MMSI;
                            Long = sdata.recordArray{ ridx, tidx }.Long;
                            Lat = sdata.recordArray{ ridx, tidx }.Lat;
                            time = sdata.recordArray{ ridx, tidx }.tsec;

                            for didx = ( ridx + 1 ) : sdata.endofData( tidx );

                                if( ~isempty( sdata.recordArray{ didx, tidx } ) )

                                    if( ( MMSI == sdata.recordArray{ didx, tidx
}.MMSI ) && ...

                                            ( Long == sdata.recordArray{ didx, tidx
}.Long ) && ...

                                            ( Lat == sdata.recordArray{ didx, tidx
}.Lat  ) && ...

                                            ( time == sdata.recordArray{ didx, tidx
}.tsec ) ) % duplicate found

                                        sdata.recordArray{ didx, tidx }  = []; %
delete record


                                    end

                                end

                            end

                        end

                    end

                end

                disp( 'Encounter detections in progress... please wait.' )

                for tidx = 1 : 86400 % for each time

                    for lidx = 1 : sdata.endofData( tidx ) % for each item in list

                        newRecord = sdata.recordArray{ lidx, tidx }; % read new
record

                        if( ~isempty( newRecord ) )

                            sdata.edetect( newRecord ); % detect encounters for this
record

                            stat.recproc = stat.recproc + 1;

                            if( mod( stat.recproc, 1000 ) == 0 )

                                disp( '-----------------------------------------
------------------------' )
                                disp( [ 'Record #: ' num2str( stat.recproc ) ' of '
num2str( recordcount ) ] )
                                disp( '-----------------------------------------
------------------------' )
                                disp(   '-----------------------------------------
```

```
--------' )
                                    disp( [ 'ENCOUNTERS = ' num2str(
sdata.totalNumEncounters ) ] )
                                    disp( [ 'GROUPS = ' num2str( sdata.numEncounterGroups
) ] )
                                    disp(   '-----------------------------------------
--------' )

                                    totaltime = toc( totalstart );
                                    disp( [ 'Time elapsed in seconds for config #: '
num2str( config ) ' = ' num2str( totaltime ) ] )

                                end

                            end

                        end

                    end

                    if( sdata.totalNumEncounters > 0 )

                        stat.totalenc = stat.totalenc + sdata.totalNumEncounters;
                        stat.totalgrp = stat.totalgrp + sdata.numEncounterGroups;
                        disp(   '-------------------------------------------------' )
                        disp( [ 'TOTAL ENCOUNTERS = ' num2str( stat.totalenc ) ] )
                        disp( [ 'TOTAL GROUPS = ' num2str( stat.totalgrp ) ] )
                        disp(   '-------------------------------------------------' )
                        encounterset = encounterset + 1;
                        edata.encounterlist{ encounterset } = sdata.encounterlist;

                    end
                end

            end
        end
    end
end



    disp(   '-------------------------------------------------' )
    disp( [ 'TOTAL ENCOUNTERS = ' num2str( stat.totalenc ) ] )
    disp( [ 'TOTAL GROUPS = ' num2str( stat.totalgrp ) ] )
    disp(   '-------------------------------------------------' )
    clear sdata

end
```

mergeEncounters.m (spatial window version)

```
% Michael Hanna
% Naval Postgraduate School
% Thesis Date June 2015

% mergeEncounters identifies and removes duplicate encounters and groups
% This is the spatial window version
% The spatial window version of encounterDetect.m is also required.

function [ sdata ] = mergeEncounters( edata, config, totalstart )

    % merge encounterlists
    sdata = AISData; % create object
    encgroupsidx = 0;
    encgroups = [];
    colend = zeros( 1, 10000 );

    [ row, col ] = size( edata.encounterlist );
    encounterset = col;
```

```
    for idx = 1 : encounterset


        [ row, col, z ] = size( edata.encounterlist{ idx } );

        if( mod( idx, 1000 ) == 0 )

            disp( [ 'Merging encounters from set #: ' num2str( idx ) ' of ' num2str(
encounterset ) ] )

        end

        for ridx = 1 : row % each encounter group

            found_group = false;
            empty_flag = true;
            colidx = 0;

            for eidx = 1 : encgroupsidx; % each existing indexed group

                MMSIi = edata.encounterlist{ idx }{ ridx, 1, 1 }.MMSI;
                MMSIj = edata.encounterlist{ idx }{ ridx, 1, 2 }.MMSI;
                if( ( ( encgroups{ eidx }.MMSIi == MMSIi ) && ( encgroups{ eidx }.MMSIj
== MMSIj ) ) || ...
                        ( ( encgroups{ eidx }.MMSIi == MMSIj ) && ( encgroups{ eidx
}.MMSIj == MMSIi ) ) ) % matches index

                    while( ( ~isempty( edata.encounterlist{ idx }{ ridx, colidx + 1, 1 }
) ) )

                        empty_flag = false;
                        colend( eidx ) = colend( eidx ) + 1;
                        colidx = colidx + 1;

                        sdata.encounterlist{ eidx, colend( eidx ), 1 } =
edata.encounterlist{ idx }{ ridx, colidx, 1 };
                        sdata.encounterlist{ eidx, colend( eidx ), 2 } =
edata.encounterlist{ idx }{ ridx, colidx, 2 };
                        sdata.encounterlist{ eidx, colend( eidx ), 3 } =
edata.encounterlist{ idx }{ ridx, colidx, 3 };

                        found_group = true;

                        if( colidx == col )

                            break

                        end

                    end

                    break

                end

            end

            if( found_group == false ) % create new entry in index and store encounters

                if( ~isempty( edata.encounterlist{ idx }{ ridx, colidx + 1, 1 } ) )

                    while( ( ~isempty( edata.encounterlist{ idx }{ ridx, colidx + 1, 1 }
) ) )

                        if( empty_flag == true )

                            encgroupsidx = encgroupsidx + 1;

                        end
```

```
                        empty_flag = false;
                        colend( encgroupsidx ) = colend( encgroupsidx ) + 1;
                        colidx = colidx + 1;

                        sdata.encounterlist{ encgroupsidx, colend( encgroupsidx ), 1 } =
edata.encounterlist{ idx }{ ridx, colidx, 1 };
                        sdata.encounterlist{ encgroupsidx, colend( encgroupsidx ), 2 } =
edata.encounterlist{ idx }{ ridx, colidx, 2 };
                        sdata.encounterlist{ encgroupsidx, colend( encgroupsidx ), 3 } =
edata.encounterlist{ idx }{ ridx, colidx, 3 };

                        encgroups{ encgroupsidx }.MMSIi = sdata.encounterlist{
encgroupsidx, 1, 1 }.MMSI;
                        encgroups{ encgroupsidx }.MMSIj = sdata.encounterlist{
encgroupsidx, 1, 2 }.MMSI;

                        if( colidx == col )

                            break

                        end

                    end

                end

            end

        end
    end

    % for each encounter list remove duplicates

    totalencountercount = 0;
    [ row, col, z ] = size( sdata.encounterlist );

    for ridx = 1 : row % for each group

        disp( [ 'Searching for duplicate encounters in group #: ' num2str( ridx ) ] )

        deleted = 0;

        for cidx = 1 : ( col - 1 )

            if( ~isempty( sdata.encounterlist{ ridx, cidx, 1 } ) )

                MMSIi = sdata.encounterlist{ ridx, cidx, 1 }.MMSI;
                MMSIj = sdata.encounterlist{ ridx, cidx, 2 }.MMSI;
                Longi = sdata.encounterlist{ ridx, cidx, 1 }.Long;
                Longj = sdata.encounterlist{ ridx, cidx, 2 }.Long;
                Lati = sdata.encounterlist{ ridx, cidx, 1 }.Lat;
                Latj = sdata.encounterlist{ ridx, cidx, 2 }.Lat;
                timei = sdata.encounterlist{ ridx, cidx, 1 }.tsec;
                timej = sdata.encounterlist{ ridx, cidx, 2 }.tsec;

                for didx = ( cidx + 1 ) : col

                    if( ~isempty( sdata.encounterlist{ ridx, didx, 1 } ) )

                        if( ( ( ( MMSIi == sdata.encounterlist{ ridx, didx, 1 }.MMSI ) &&
...
                                ( Longi == sdata.encounterlist{ ridx, didx, 1 }.Long ) &&
...
                                ( Lati == sdata.encounterlist{ ridx, didx, 1 }.Lat ) &&
...
                                ( timei == sdata.encounterlist{ ridx, didx, 1 }.tsec ) )
&& ...
                                ( ( MMSIj == sdata.encounterlist{ ridx, didx, 2 }.MMSI )
```

```
&& ...
                                ( Longj == sdata.encounterlist{ ridx, didx, 2 }.Long ) &&
...
                                ( Latj == sdata.encounterlist{ ridx, didx, 2 }.Lat ) &&
...
                                ( timej == sdata.encounterlist{ ridx, didx, 2 }.tsec ) )
) || ...
                                ( ( ( MMSIi == sdata.encounterlist{ ridx, didx, 2 }.MMSI
) && ...
                                ( Longi == sdata.encounterlist{ ridx, didx, 2 }.Long ) &&
...
                                ( Lati == sdata.encounterlist{ ridx, didx, 2 }.Lat ) &&
...
                                ( timei == sdata.encounterlist{ ridx, didx, 2 }.tsec ) )
&& ...
                                ( ( MMSIj == sdata.encounterlist{ ridx, didx, 1 }.MMSI )
&& ...
                                ( Longj == sdata.encounterlist{ ridx, didx, 1 }.Long ) &&
...
                                ( Latj == sdata.encounterlist{ ridx, didx, 1 }.Lat ) &&
...
                                ( timej == sdata.encounterlist{ ridx, didx, 1 }.tsec ) )
) ) % duplicate found

                            % disp( 'Duplicate found. Deleted.' )
                            sdata.encounterlist{ ridx, didx, 1 } = []; % delete encounter
                            sdata.encounterlist{ ridx, didx, 2 } = [];
                            sdata.encounterlist{ ridx, didx, 3 } = 0;
                            deleted = deleted + 1;

                        end

                    end

                end

            end

        end

        encountercount = 0;

        disp( 'counting encounters...' )

        for idxcnt = 1 : col

            if( ~isempty( sdata.encounterlist{ ridx, idxcnt, 1 } ) )

                encountercount = encountercount + 1;

            end

        end

        disp( [ 'Encounters found in group # ' num2str( ridx ) ' of ' num2str( row ) ': '
num2str( encountercount ) ] )
        disp( [ 'Encounters deleted: ' num2str( deleted ) ] )
        totalencountercount = totalencountercount + encountercount;

        disp( [ 'Searching for duplicate entries in group #: ' num2str( ridx ) ] )

        deleted = 0;

        for cidx = 1 : ( col - 1 )

            if( ~isempty( sdata.encounterlist{ ridx, cidx, 1 } ) )

                MMSI = sdata.encounterlist{ ridx, cidx, 1 }.MMSI;
                Long = sdata.encounterlist{ ridx, cidx, 1 }.Long;
```

```
                Lat = sdata.encounterlist{ ridx, cidx, 1 }.Lat;
                time = sdata.encounterlist{ ridx, cidx, 1 }.tsec;

                for didx = ( cidx + 1 ) : col

                    if( ~isempty( sdata.encounterlist{ ridx, didx, 1 } ) )

                        if( ( MMSI == sdata.encounterlist{ ridx, didx, 1 }.MMSI ) && ...
                                ( Long == sdata.encounterlist{ ridx, didx, 1 }.Long ) &&
...
                                ( Lat == sdata.encounterlist{ ridx, didx, 1 }.Lat  ) &&
...
                                ( time == sdata.encounterlist{ ridx, didx, 1 }.tsec ) ) %
duplicate found

                            % disp( 'Duplicate found. Deleted.' )
                            sdata.encounterlist{ ridx, didx, 1 } = []; % delete encounter
                            sdata.encounterlist{ ridx, didx, 3 } = 0;
                            deleted = deleted + 1;

                        end

                    end

                end

            end

        end

        disp( [ 'Entries deleted: ' num2str( deleted ) ] )

        deleted = 0;

        for cidx = 1 : ( col - 1 )

            if( ~isempty( sdata.encounterlist{ ridx, cidx, 2 } ) )

                MMSI = sdata.encounterlist{ ridx, cidx, 2 }.MMSI;
                Long = sdata.encounterlist{ ridx, cidx, 2 }.Long;
                Lat = sdata.encounterlist{ ridx, cidx, 2 }.Lat;
                time = sdata.encounterlist{ ridx, cidx, 2 }.tsec;

                for didx = ( cidx + 1 ) : col

                    if( ~isempty( sdata.encounterlist{ ridx, didx, 2 } ) )

                        if( ( MMSI == sdata.encounterlist{ ridx, didx, 2 }.MMSI ) && ...
                                ( Long == sdata.encounterlist{ ridx, didx, 2 }.Long ) &&
...
                                ( Lat == sdata.encounterlist{ ridx, didx, 2 }.Lat  ) &&
...
                                ( time == sdata.encounterlist{ ridx, didx, 2 }.tsec ) ) %
duplicate found

                            % disp( 'Duplicate found. Deleted.' )
                            sdata.encounterlist{ ridx, didx, 2 } = []; % delete encounter
                            sdata.encounterlist{ ridx, didx, 3 } = 0;
                            deleted = deleted + 1;

                        end

                    end

                end

            end

        end
```

```
        disp( [ 'Entries deleted: ' num2str( deleted ) ] )
                totaltime = toc( totalstart );
        disp( [ 'Time elapsed in seconds for config #: ' num2str( config ) ' = ' num2str(
totaltime ) ] )

    end

    disp( [ 'Total Encounters found: ' num2str( totalencountercount ) ] )

    sdata.totalencounters = totalencountercount;
    sdata.groups = row;
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] K. L. Barber, "NSG expeditionary architecture: harnessing big data," *National Geospatial-Intelligence Agency Magazine: Pathfinder,* vol. 10, no. 5, pp. 8–10, Sept./Oct. 2012.

[2] T. D. Lash, "Uses of motion imagery in activity-based intelligence," *Proc. SPIE,* 2013, vol. 8740, 874005 (May 16, 2013).

[3] C. Johnston (2013). *Modernizing defense intelligence: object based production and activity based intelligence* [PowerPoint slides]. Retrieved from https://info.publicintelligence.net/DIA-ActivityBasedIntelligence.pdf," 2013.

[4] D. Meyerriecks (2012). *Empowering intelligence integration: the (future) role of ground* [PowerPoint slides]. Retrieved from http://gsaw.org/wp-content/uploads/2013/07/2012s08meyerriecks.pdf

[5] W. Raetz, "A new approach to graph analysis for activity based intelligence," *Proc. IEEE Applied Imagery Pattern Recognition Workshop,* Washington, DC, pp. 1–8, 2012.

[6] U.S. Department of Homeland Security (DHS), "National plan to achieve maritime domain awareness for the national strategy for maritime security," DHS, Washington, DC, Oct. 2005.

[7] D. A. Goward, "Maritime domain awareness integration challenges," United States Coast Guard, Washington, DC, 2008.

[8] K. A. Tester, "A spatiotemporal clustering approach to maritime domain awareness," M.S. thesis, Dept. Electrical and Computer Eng., Naval Postgraduate School, Monterey, CA, 2013.

[9] A. S. McAbee, "Traffic pattern detection using the hough transformation for anomaly detection to improve maritime domain awareness," M.S. thesis, Dept. Electrical and Computer Eng., Naval Postgraduate School, Monterey, CA, 2013.

[10] J. J. Thomas and K. A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* Los Alamitos, CA: IEEE, 2005.

[11] G. Andrienko, N. Andrienko, S. Wrobel, P. Bak and D. Keim, *Visual Analytics of Movement.* New York: Springer, 2013.

[12] M. Riveiro, "Visual analytics for maritime domain detection," Ph.D. dissertation, Studies in Technology 46, Orebro Univ., Orebro, Sweden, 2011.

[13] Y. Fischer and A. Bauer, "Object-oriented sensor data fusion for wide maritime surveillance," *Proc. Intl. Waterside Security Conference,* pp. 1–6, 2010.

[14] R. Scheepens, N. Willems, H. Van De Wetering and J. J. van Wijk, "Interactive visualization of multivariate trajectory data with density maps," in *Proc. IEEE Pacific Visualization Symposium*, Hong Kong, China, pp. 147–154, 2011.

[15] C. Tominski, H. Schumann, G. Andrienko and N. Andrienko, "Stacking-Based Visualization of Trajectory Attribute Data," *IEEE Transactions on Visualization and Computer Graphics,* vol. 18, pp. 2565–2574, 2012.

[16] M. Davenport, "Literature and product review of visual analytics for maritime awareness," Ottawa, ON: Defense Research and Development Canada, Oct. 2009.

[17] P. v. de Laar, J. Tretmans, M. Borth and Embedded Systems Institute, *Situation Awareness with Systems of Systems.* New York: Springer, 2013.

[18] J. Llinas and J. Scrofani, "Foundational Technologies for Activity-based intelligence: a review of the literature ," Naval Postgraduate School, Monterey, CA, Feb. 2014.

[19] M. Phillips, (2012, Sept. 28). A brief overview of activity based intelligence and human domain analytics, Trajectory [Online]. http://trajectorymagazine.com/defense-intelligence/item/1369-human-domain-analytics.html

[20] K. Quinn, (Winter 2012). A better toolbox: analytic methodology has evolved significantly since the cold war, Trajectory [Online]. http://trajectorymagazine.com/civil/item/1349-a-better-toolbox.html

[21] G. Miller, (2013, July 8). Activity-based intelligence uses metadata to map adversary networks, *Defense News* [Online]. http://archive.defensenews.com/article/20130708/C4ISR02/307010020/Activity-Based-Intelligence-Uses-Metadata-Map-Adversary-Networks

[22] D. B. Cousins, D. J. Weishar and J. B. Sharkey, "Intelligence collection for counter terrorism in massive information content," in *Proc. Aerospace Conference,* vol. 5, pp. 3273–3282, 2004.

[23] G. Andrienko, N. Andrienko and U. Demsar, "Space, time and visual analytics," *Intl. Journal of Geographical Information Science*, vol. 24, no. 10, pp. 1577–1600, 2010.

[24] E. Blasch, A. Steinberg, S. Das, J. Llinas, Chee Chong, O. Kessler, E. Waltz and F. White, "Revisiting the JDL model for information exploitation," in *16th International Conference Information Fusion,* Istanbul, Turkey, pp. 129–136, 2013.

[25] M. Ward, G. G. Grinstein and D. Keim, *Interactive Data Visualization: Foundations, Techniques, and Applications.* Natick, MA: A K Peters, 2010.

[26] C. Ware, *Information Visualization: Perception for Design.* Waltham, MA: Morgan Kaufmann, 2013.

[27] I. H. Witten, E. Frank and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques.* Burlington, MA: Morgan Kaufmann, 2011.

[28] T. H. Cormen, *Algorithms Unlocked.* Cambridge, Massachusetts: The MIT Press, 2013.

[29] P. Laube, M. Kreveld and S. Imfeld, "Finding REMO - detecting relative motion patterns in geospatial lifelines," *Developments in Spatial Data Handling,* P. Fisher, Ed., New York,: Springer, pp. 201–215, 2005.

[30] P. Bak, M. Marder, S. Harary, A. Yaeli and H. Ship, "Scalable detection of spatiotemporal encounters in historical movement data," in *Computer Graphics Forum,* vol. 31, no. 3, pp. 915–924, 2012.

[31] S. Dodge, R. Weibel and A. Lautenschutz, "Towards a taxonomy of movement patterns," *Information Visualization,* vol. 7, no. 3, pp. 240–252, 2008.

[32] G. Andrienko, N. Andrienko and M. Heurich, "An event-based conceptual model for context-aware movement analysis," *Intl. Journal of Geographical Information Science,* vol. 25, no. 9, pp. 1347–1370, 2011.

[33] D. Orellana, M. Wachowicz, N. Andrienko and G. Andrienko, "Uncovering interaction patterns in mobile outdoor gaming," *Intl. Conf. on Adv. Geographic Information Systems & Web Services,* Cancun, Mexico, pp. 177–182, 2009.

[34] G. Andrienko, N. Andrienko, P. Bak and D. Keim, "A conceptual framework and taxonomy of techniques for analyzing movement," *Journal of Visual Languages and Computing*, vol. 22, no. 3, pp. 213–232, 2011.

[35] J. Gudmundsson, M. v. Kreveld and B. Speckmann, "Efficient detection of patterns in 2D trajectories of moving points," *GeoInformatica,* vol. 11, no. 2, pp. 195–215, 2007.

[36] T. H. Cormen, *Introduction to Algorithms.* Cambridge, Mass.: MIT Press, 2009.

[37] N. Arundale, AisDecoder. [Online]. Available: http://nmearouter.com/docs/ais/ais_decoder_v3_downloads.html

[38] Raymond, E. S., AIVDM/AIVDO protocol decoding. [Online]. Available: http://catb.org/gpsd/AIVDM.html

[39] G. Andrienko, V-Analytics. [Online]. Available: http://geoanalytics.net/V-Analytics

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California